*b) (**5 puntos**) En el instante t=0 s la cabeza se encuentra situada al comienzo del sector absoluto 0. En ese instante se ordena la lectura de un fichero que ocupa 205.824 bytes. El comienzo del fichero reside en los sectores 3.201.720 y 7.680.921, estando el resto almacenado en sectores consecutivos a partir del 4.483.042. Calcule el instante en el que finaliza la lectura de dicho fichero.*

SOLUCIÓN

a) Número de cilindros:

$\frac{81{,}92 \cdot 10^9 \text{ bytes} \times 0{,}8}{512 \times 200 \times 16} = 40.000$ cilindros

Velocidad de transferencia:

$\frac{512}{0{,}8} \times 200 \times \frac{12.000}{60} = 25{,}6 \cdot 10^6$ bytes/s

b) Tiempo de giro: 60/12.000 = 0,005 s = 5 ms. Tiempo de avance de un sector: 5/200 = 0,025 ms.

Número de sectores del fichero: 205.824 / 512 = 402 sectores.

Coordenadas geométricas (CHS) de los sectores:

3.201.720/(16 · 200) = 1.000 (resto 1.720)

1.720/200 = 8; resto 120

3.201.720 → (1.000, 8, 120)

Procediendo de igual forma para los demás:

7.680.921 → (2.400, 4, 121)

4.483.042 → (1.400, 15, 42)

Sector 3.201.720:

Tbúsqueda = (1.000 - 0)0,01 + 1 = 11 ms (2 vueltas completas más 1 ms durante el cual avanza 1/0,025 = 40 sectores, quedando las cabezas delante del sector 40).

Tlatencia = (120 - 40)0,025 = 2 ms

Lectura = 0,025

Total: 11 + 2 + 0,025 = 13,025 ms

Sector 7.680.921:

Tbúsqueda = (2.400 - 1.000)0,01 + 1 = 15 ms (3 vueltas completas, quedando las cabezas delante del sector 121).

Tlatencia = 0

Lectura = 0,025 ms

Total : 15 + 0,025 = 15,025 ms

Sector 4.483.042 y siguientes:

Tbúsqueda = (2.400 -1.500)0,01 + 1 = 11 ms (2 vueltas completas más 1 ms durante el cual avanza 1/0,025 = 40 sectores, quedando las cabezas delante del sector 122 + 40 = 162).

Tlatencia = (200 - 162 + 42)0,025 = 2 ms.

Los siguientes 158 sectores son consecutivos, están en la última pista del cilindro 1.500 y tienen un tiempo de acceso nulo. Para acceder al siguiente sector es preciso mover las cabezas al siguiente cilindro (1.401) empleando 1,01 ms y esperar a que el disco complete una vuelta, quedando nuevamente sobre el sector 0. En total, el tiempo de acceso del siguiente sector son 5 ms. Los últimos 242 sectores (400 - 158) son consecutivos, pertenecen al mismo cilindro y, nuevamente, tienen un tiempo de acceso nulo. El tiempo de lectura de los últimos 400 sectores del fichero son 15 ms (400·0,025 ms + 5 ms correspondientes al cambio de cilindro).

Total = 11 + 2 + 15 = 28 ms

Tiempo total de lectura del fichero:

13,025 + 15,025 + 28 = 56,05 ms

1 *(**1 punto**) Sea una cinta magnética con las siguientes características:*

- *Tiempo de arranque y de parada: 2 ms.*
- *Velocidad de transferencia: 32 Mbytes/s (32×10^6 bytes/s).*
- *Claros IRG de 0,2 cm de longitud.*
- *Densidad de grabación lineal: 32.000 bits/mm.*
- *Bloques de 16.000 bytes.*

a) Calcule cuánto tiempo se emplea en arrancar la cinta, leer dos bloques consecutivos y detener la cinta.

SOLUCIÓN

$$2 \text{ ms} + \frac{16.000+0{,}2\times10\times32.000/8+16.000}{32\times10^6} \text{ s} + 2 \text{ ms} = 5{,}25 \text{ ms}$$

2 *(**1 punto**) Dado un monitor LCD con una resolución de 1.920×1.200 píxeles y 70 Hz de frecuencia de refresco que se conecta a una placa gráfica cuya memoria de pantalla accede a palabras de 32 bits con un tiempo de acceso de 8 ns,*

a) calcule cuál es la máxima profundidad de color con que se puede configurar el monitor.

SOLUCIÓN

a) Requisitos del monitor:

$1.920 \times 1.200 \times 70 = 161.280.000$ píxeles/s

Capacidad de la placa gráfica:

$\frac{32 \text{ bits}}{8\times10^{-9} \text{ s}} = 4.000.000.000$ bits/s

$\frac{4.000.000.000 \text{ bits/s}}{161.280.000 \text{ píxeles/s}} = 24{,}8$ bits/píxel

3 *Sea una unidad de disco duro con las siguientes características:*

- *Capacidad Bruta: 172.800.000.000 bytes.*
- *Capacidad Neta: 147.456.000.000 bytes.*
- *18 superficies y 400 sectores por pista.*
- *Sectores de 1.024 bytes de información neta.*
- *Velocidad de transferencia: 60 MB/s ($60 \cdot 10^6$ bytes/s).*
- *Tiempo que emplea en mover la cabeza de una pista a otra consecutiva: 0,02 ms.*
- *Tiempo de estabilización de las cabezas: 2,5 ms.*

*a) (**2 puntos**) Calcule el número de cilindros y la velocidad rotación de la unidad de disco.*

*b) (**5 puntos**) En el instante t=0 s la cabeza se encuentra situada al comienzo del sector absoluto 0. En ese instante se ordena la lectura de un fichero que está almacenado en los siguientes sectores: 7.203.320, 7.203.319, 7.203.318, 2.347.199, 2.347.200 y 2.347.201. Calcule el instante en el que finaliza la lectura de dicho fichero.*

*c) (**1 punto**) Una vez leído todo el fichero, se envía por una línea serie asíncrona, sin paridad y con 1 bit de stop empleando 3,2 segundos. Calcule la velocidad de transmisión de la línea serie.*

SOLUCIÓN

a) Número de cilindros:

$\frac{147.456.000.000 \text{ bytes}}{1.024\times400\times18 \text{ bytes/cilindro}} = 20.000$ cilindros

Velocidad de rotación:

$\frac{172.800.000.000 \text{ bytes}}{20.000 \times 18 \text{ pistas}} = 480.000$ bytes/pista

$\frac{60 \cdot 10^6 \text{ bytes/s}}{480.000 \text{ bytes/pista}} = 125$ pistas/s = 125 r.p.s = 125×60 r.p.m. = 7.500 r.p.m.

b) Tiempo de giro: 1/125 = 0,008 s = 8 ms. Tiempo de avance de un sector: 8/400 = 0,02 ms.

Coordenadas geométricas (CHS) de los sectores:

7.203.320/(18 · 400) = 1.000 (resto 3.320)

3.320/400 = 8; resto 120

7.203.320 → (1.000, 8, 120)

7.203.319 → (1.000, 8, 119)

7.203.318 → (1.000, 8, 118)

Procediendo de igual forma para los demás:

2.347.199 → (325, 17, 399)

2.347.200 → (326, 0, 0)

2.347.201 → (326, 0, 1)

Sector 7.203.320:

Tbúsqueda = (1.000 - 0)0,02 + 2,5 = 22,5 ms (2 vueltas completas más 6,5 ms durante los cuales avanza 6,5/0,02 = 325 sectores, quedando las cabezas delante del sector 325).

Tlatencia = (400 - 325 + 120)0,02 = 3,9 ms

Lectura = 0,02 ms

Total: 22,5 + 3,9 + 0,02 = 26,42 ms

Sector 7.203.319:

Tbúsqueda = 0

Tlatencia = (400 - 121 + 119)0,02 = 7,96 ms

Lectura = 0,02 ms

Total : 0 + 7,96 + 0,02 = 7,98 ms

Sector 7.203.318:

Tbúsqueda = 0

Tlatencia = (400 - 120 + 118)0,02 = 7,96 ms

Lectura = 0,02 ms

Total : 0 + 7,96 + 0,02 = 7,98 ms

Sector 2.347.199 y siguientes:

Tbúsqueda = (1.000 - 325)0,02 + 2,5 = 16 ms (2 vueltas completas, quedando las cabezas delante del sector 119).

Tlatencia = (399 - 119)0,02 = 5,6 ms.

Lectura = 0,02 ms

Total : 16 + 5,6 + 0,02 = 21,62 ms

Para acceder al siguiente sector (2.347.200) es preciso mover las cabezas al siguiente cilindro (326) empleando 2,52 ms y esperar a que el disco complete una vuelta, quedando nuevamente sobre el sector 0. En total, el tiempo de acceso son 8 ms. El último sector (2.347.201) es consecutivo, pertenece al mismo cilindro y tiene un tiempo de acceso nulo. El tiempo de lectura de estos dos últimos sectores del fichero son 8,04 ms (2·0,02 ms + 8 ms correspondientes al cambio de cilindro).

Total = 8 + 0,02 + 0,02 = 8,04 ms

Tiempo total de lectura del fichero:

26,42 + 7,98 +7,98 + 21,62 + 8,04 = 72,04 ms

c) El fichero ocupa 6 sectores y tiene una longitud de $6 \times 1.024 = 6.144$ bytes. Por cada byte se transmiten 10 bits (1 bit de start, 8 bits de dato y 1 bit de stop) dando un total de 61.440 bits. Dado que se emplean 3,2 s, la velocidad de transmisión es $61.440/3{,}2 = 19.200$ bits/s.

1 **(3 puntos)** *Programe en ensamblador del 88110 los fragmentos de la subrutina* `funcion` *que se detallan a continuación:*

```
int funcion (int p1, int p2, int vector[])
{
   /* Fragmento 1 */

   /* Variables locales enteras */
   int  i, j;
   /* Matriz de enteros de 7 filas y 7 columnas almacenada por filas*/
   int matriz [7][7];

   /* Se inicializan todos los elementos de la matriz con su número de fila */
   for (i=0; i < 7; i = i + 1) {
       for (j=0; j < 7; j = j + 1) {
           matriz[i][j] = i;
       }
   }

       ....

   /* Fragmento 2 */

   i = seleccionar (matriz, vector);
   j = manipular (matriz, i, p1);
   return i+j; // El valor de retorno (i+j) se devuelve en r29
   /* Fin */
}
```

a) Fragmento 1: Creación del marco de pila incluyendo la inicialización de la variable local `matriz`*.*

b) Fragmento 2: Llamada a la subrutina `seleccionar` *incluyendo el paso de los parámetros por dirección y en la pila. Llamada a la subrutina* `manipular`*, que recibe el primer parámetro por dirección y los dos últimos por valor. Suponga que la subrutina llamante,* `funcion`*, deja disponibles todos los registros excepto r1, r30 (SP) y r31 (FP) y que todas las subrutinas del enunciado devuelven el valor de retorno en r29.*

SOLUCIÓN

En el código del enunciado hay que crear dos variables enteras que están en las posiciones -4 y -8 frente a r31. Además hay que crear una matriz de 49 palabras que comienza donde apunta r30. El código comentado se muestra a continuación:

```
// Fragmento 1
funcion: PUSH(r1)
         PUSH(r31)
         or r31,r30,r30
         or r4,r0,7
         mulu r4,r4,r4
         mulu r4,r4,4
         subu r30,r30,8       ; Se reservan dos palabras para i y j
         subu r30,r30,r4      ; Se reservan 49 palabras para matriz
         or r3,r0,r0          ; fila
         or r4,r0,r0          ; columna
         or r20,r30,r0        ; Puntero al elemento de la matriz

bucle:   st r3,r20,r0         ; Se almacena i en el elemento de la matriz
         addu r4,r4,1         ; Se incrementa la columna
         addu r20,r20,4
         cmp r5,r4,7
         bb1 ne,r5,bucle
```

```
        or r4,r0,r0          ; Se pone a 0 la columna
        addu r3,r3,1         ; Se incrementa la fila
        cmp r5,r3,7
        bb1 ne,r5,bucle


// Fragmento 2
        ld r20,r31,16
        PUSH(r20)            ; Se pasa vector
        or r20,r30,r0
        PUSH(r20)            ; Se pasa matriz
        bsr seleccionar
        st r29,r31,-4        ; Se almacena i
        addu r30,r30,8
        ld r20,r31,8
        PUSH(r20)            ; Se pasa p1
        PUSH(r29)            ; Se pasa i
        or r20,r30,r0
        PUSH(r20)            ; Se pasa matriz
        bsr manipular
        addu r30,r30,12
        ld r5,r31,-4
        addu r29,r29,r5      ; Al valor de retorno de manipular (r29) se le suma i
        or r30,r31,r31       ; Este es el valor de retorno de funcion (r29)
        POP(r31)
        POP(r1)
        jmp(r1)
```

2 *(2 puntos) Sea una unidad de disco duro de brazo móvil con las siguientes características:*

- *16 superficies, 9.901 cilindros y 200 sectores por pista.*
- *Sectores de 1.250 bytes de información bruta y 1.024 bytes de información neta.*
- *Velocidad de rotación: 6.000 rpm.*
- *Velocidad de transferencia: 25 MB/s ($25 \cdot 10^6$ bytes/s).*
- *Tiempo que emplea en mover la cabeza de una pista a otra consecutiva: 0,15 ms.*
- *Tiempo de estabilización de las cabezas: 2,5 ms.*

Dicha unidad de disco duro contiene un fichero de 30.720 bytes cuyos primeros datos están almacenados en los sectores 10.628.880 y 10.948.881. El resto de los datos del fichero está almacenado en sectores distribuidos aleatoriamente por el disco.

a) Suponiendo que la posición inicial de las cabezas de lectura es Cilindro 4.321, Sector 100 y que el tiempo de cómputo es despreciable, calcule el tiempo que se emplea en leer los dos primeros sectores del fichero.

b) Calcule cuál es el tiempo medio de lectura del fichero completo.

Una vez finalizada la lectura del fichero, se envía éste a un grabador de discos compactos, CD-WR, que opera a una velocidad 10x y tiene un formato de grabación con sectores de 2.048 bytes netos. Para ello cada sector se completa con 13 bytes de sincronismo, 3 bytes de cabecera y 288 bytes de códigos de control de errores formando sectores de 2.352 bytes de información bruta. Los sectores así formados se envían a la unidad CD-RW a una velocidad sostenida de 1,764 MB/s ($1{,}764 \cdot 10^6$ bytes/s).

c) Suponiendo que la transferencia de los datos del fichero comienza 2 segundos después de finalizada su lectura y que se realiza mediante una única operación de E/S, calcule en qué instante finaliza la transferencia de todo el fichero.

SOLUCIÓN

a) Sector_Absoluto = Cilindro · (200 · 16) + Superficie · 200 + Sector.

10.628.880 = 3.321 · 3.200 + 8 · 200 + 80 ⇒ Cilindro 3.221, Superficie 8, Sector 80.

10.948.881 = 3.421 · 3.200 + 8 · 200 + 81 ⇒ Cilindro 3.421, Superficie 8, Sector 81.

Tiempo que se tarda en dar una vuelta completa = 60/6.000 s = 10 ms.

Tiempo que se tarda en recorrer un sector = 10/200 = 0,05 ms.

Sector 10.628.880 (3.321, 8, 80):

- Tbusqueda = (4.321 - 3.321)·0,15 + 2,5 = 152,5 ms.
- Sectores recorridos durante el Tbusqueda: 152,5/0,05 = 3.050.
- Posición actual (100 + 3.050) mod 200 = 150.
- Tlatencia = (200 - 150 + 80) · 0,02 = 6,5 ms.
- Tlectura = 0,05 ms.
- Tiempo empleado el leer el sector 10.628.880 = 152,5 + 6,5 + 0,05 = 159,05 ms.

Sector 10.948.881 (3.421, 8, 81):

- Tbusqueda = (3.421 - 3.321) · 0,15 + 2,5 = 17,5 ms.
- Sectores recorridos durante el Tbusqueda: 17,5/0,05 = 350.
- Posición actual (81 + 350) mod 200 = 31.
- Tlatencia = (81 + 31) · 0,05 = 2,5 ms.
- Tlectura = 0,05 ms.
- Tiempo empleado el leer el sector 10.948.881 = 17,5 + 2,5 + 0,05 = 20,05 ms.

Tiempo empleado en leer los dos primeros sectores = 159,05 + 20,05 = 179,1 ms.

b) Tiempo medio de acceso de la unidad de disco:

Tmacc = 1/2 · (9.901 - 1) · 0,15 ms + 2,5 ms + 1/2 · 60/6.000 s = 742,5 ms + 2,5 ms + 5 ms = 750 ms.

El fichero ocupa 30.720/1.024 = 30 sectores. Teniendo en cuenta el tiempo medio de acceso calculado, el tiempo medio que se tarda en la lectura de un sector es 750 + 0,05 = 750,05 ms. Por tanto, la lectura del fichero se completa en el instante t = 179,1 ms + 28 · 750,05 = 21.180,5 ms.

c) La unidad de CD-RW usa sectores de 2.048 bytes netos, por lo que el fichero de 30.720 bytes ocupará 15 sectores (30.720/2.048 = 15). La información bruta correspondiente al fichero es 15 · 2.352 = 35.280 bytes que, a la velocidad de transferencia de 1,764 MB/s, tardan $35.280/1{,}764 \cdot 10^6$ s = 0,02 s. La transferencia finaliza en el instante t = 21.180,5 ms + 2.000 ms + 20 ms = 23.200,5 ms.

3 *(3 puntos) Dados los números decimales A = 146.5 B = - 0.75 C = - 511*

a) Obtenga de forma razonada su representación en los siguientes formatos numéricos:

- *Formato 1: Coma fija en complemento a 2 con 10 bits de parte entera y 6 bits de parte fraccionaria.*
- *Formato 2: Coma fija en signo magnitud con el primer bit para el signo. 9 bits de parte entera y 6 bits de parte fraccionaria.*
- *Formato 3: Coma flotante con mantisa en signo-magnitud y exponente en exceso a 64, siendo el primer bit para el signo, los siete siguientes para el exponente y los ocho últimos para la magnitud de la mantisa. Dispone de bit implícito con la coma a la derecha de éste.*

b) Realice justificadamente la operación A - C en cada uno de los dos formatos de coma fija.

c) Realice detalladamente la operación A - C en el formato de coma flotante y obtenga el valor decimal del resultado. Utilice dos bits de guarda, un bit retenedor y redondeo al más próximo.

SOLUCIÓN

a) Representación de los números en los tres formatos.

Formato 1: Coma fija, complemento a 2

$A = 146,5 = 0010010010,100000$

$B = -0,75 = -\ 0000000000,110000 =$ (*complementando a* 2) $1111111111,010000$

$C = -511 = -\ 0111111111,000000 =$ (*complementando a* 2) $1000000001,000000$

Formato 2: Coma fija, signo magnitud

$A = 0010010010,100000$

$B = 1000000000,110000$

$C = 1111111111,000000$

Formato 3: Coma flotante

$A = 1,00100101 \cdot 2^7 = 0\ 1000111\ 00100101$

$B = -\ 1,10000000 \cdot 2^{-1} = 1\ 0111111\ 10000000$

$C = -\ 1,11111111 \cdot 2^8 = 1\ 1001000\ 11111111$

b) Operación A - C en los formatos de coma fija.

Formato 1: Coma fija, complemento a 2

$A - C = 0010010010,100000 - 1000000001,000000 = 0010010010,100000 + 0111111111,000000 =$
$= 1010010001,100000$ Desbordamiento

Formato 2: Coma fija, signo magnitud

$A - C = A - (-C) = A + C$

Puesto que los signos de los números a sumar son iguales, el signo del resultado es el signo común y la operación a realizar es la suma de las magnitudes de ambos números.

```
 010010010,100000
 111111111,000000
-----------------
1010010001,100000
```

Desbordamiento.

c) Operación A - C en el formato de coma flotante.

Los números a operar son: $A = 1,00100101 \cdot 2^7$ y $C = -\ 1,11111111 \cdot 2^8$

Como los exponentes son distintos, el exponente del resultado es: $E = E_C = 8$.

Se restan los exponentes: $E_C - E_A = 1$.

Hay que desplazar la mantisa de A un lugar a la derecha, utilizando dos bits de guarda y un bit retenedor:

$A = 0,10010010\ 10\ 0 \cdot 2^8$

$A - C = A - (-C) = A + C$

Puesto que los signos de los números a sumar son iguales, el signo del resultado es el signo común y la operación a realizar es la suma de las magnitudes de las mantisas de ambos números.

```
  0,10010010  10  0
+ 1,11111111  00  0
-----------------
 10,10010001  10  0     No normalizado.
  1,01001000  11  0     Normalización. Incremento del exponente: E = 9
+             1         Redondeo
-----------------
  1,01001001  0
```

$A - C = 1{,}010010010 \cdot 2^9 = 1010010010 = 658.$

4 *(2 puntos) Un procesador de 32 bits con unidad de control cableada, un solo bus interno de datos y direcciones y dos registros transparentes, presenta los siguientes retardos:*

Lectura o escritura del Banco de registros: 5 unidades de tiempo (ut)

Lectura o escritura de registros transparentes o específicos: 2 ut

Multiplexores de la ALU: 3 ut

Operación de mayor duración en la ALU: 8 ut

Puertas triestado: 1 ut

Así mismo, la Memoria principal se direcciona a nivel de byte y su tiempo de acceso es de 30 ut.

a) Especifique a nivel RT (transferencia entre registros) las operaciones elementales que se realizan durante las fases de fetch y de ejecución de las siguientes instrucciones de una palabra:

```
BR [.R3 ++]
ST .R5, [-- .R4]
```

b) Determine razonadamente el periodo de reloj y el tiempo total que tardarían en ejecutarse cada una de las instrucciones anteriores.

SOLUCIÓN

a) Las operaciones elementales para cada una de las dos instrucciones son las siguientes:

```
Fetch

AR<- PC
DR<- M(AR)
RI<- DR
PC<- PC + 4

Fase de ejecución instrucción BR [.R3 ++]

PC<- R3
R3<- R3 + 4

Fase de ejecución instrucción ST .R5,[-- .R4]

DR<- R5
AR, R4<- R4 - 4
M(AR)<- DR
```

b) Para determinar el periodo de reloj, se calcula el tiempo de la operación elemental de mayor duración:

$T_{ck} = t_{BR} + t_{Mx} + t_{ALU} + t_{triestado} + t_{BR} = (5 + 3 + 8 + 1 + 5)ut = 22ut$

El tiempo de ejecución de cada instrucción será

$t_{ejec} = t_{fetch} + t_{decodific} + t_{fase\ de\ ejec}$

Considerando que cada operación elemental se realiza en un ciclo de reloj y que, con el periodo de reloj obtenido, los accesos a memoria precisan dos ciclos, se tendrá

$t_{ejec\ \ BR} = ((3 + 2).22) + 22 + 2.22)ut = 176ut$

$t_{ejec\ \ ST} = ((3 + 2).22) + 22 + (2 + 2).22)ut = 220ut$

1 *Indique, justificando su respuesta, si las siguientes afirmaciones son verdaderas o falsas:*

a) La Entrada/Salida programada proporciona mejores prestaciones que por interrupciones, porque la CPU tiene el control de toda la operación.

Falso. La E/S programada proporciona peores prestaciones que por interrupciones, porque la CPU debe encargarse de la sincronización con el periférico haciendo una espera activa.

a) El Registro de Estado es un registro transparente al programador.

Falso. El programador puede referenciar el registro de estado en varias instrucciones, como los saltos condicionales, por lo que no es transparente al programador. Tiene una función específica.

a) El modelo Von Neumann se basa en una única memoria para almacenar tanto datos como instrucciones, por lo que en principio, se podría erróneamente intentar ejecutar un dato.

Verdadero. Al almacenar tanto datos como instrucciones, en el caso de que la memoria no se organice correctamente puede erróneamente intentarse un dato como si fuera una instrucción. Hay que establecer mecanismos adecuados para que eso no ocurra.

a) Las únicas entradas que necesita la Unidad de Control son: el Registro de Instrucción, el reloj, y para comunicarse con los periféricos, las señales de E/S.

Falso. Es fundamental también que la señal de reloj sea una entrada de la UC para que ésta pueda secuenciar correctamente las operaciones elementales en las que se descompone una instrucción.

2 *Sea un computador de dos direcciones con modelo de ejecución registro-registro con palabra de 32 bits y direccionamiento a nivel de byte que dispone únicamente de direccionamiento inmediato, directo a registro e indirecto a registro. Realice los programas correspondientes a las instrucciones que se muestran a continuación para el computador indicado. Si es necesario utilice los registros RT1, RT2 y RT3 como registros auxiliares.*

```
BZ  #16[++.R7]

     ADD .R7,#4
     MOVE .R7,.RT1
     ADD .RT1,#16
     BZ[.RT1]


CALL [/1000]

     LD .RT1,#1000
     LD .RT1,[.RT1]
     CALL [.RT1]


DBNZ .R1, [#16[++.R7]]        (decrementa primer operando y si NZ salta al segundo)

     ADD .R7,#4
     MOVE .R7,.RT1
     ADD .RT1,#16
     LD .RT1,[.RT1]
     DBNZ .RT1,[.RT1]


XOR [.R4],#16[.R7++]

     MOVE .R7,.RT1
     ADD .RT1,#16
     LD .RT1,[.RT1]
     ADD .R7,#4
     LD .RT2,[.R4]
     XOR .RT2, .RT1
     ST .RT2, [.R4]
```

3 *Programe en ensamblador del MC88110 las siguientes subrutinas:*

a) Subrutina `CEROS(x, vector)` *que calcula el número de elementos que son 0 en un vector de x elementos enteros de una palabra, siendo* $0 < x$. *Los parámetros se pasan en la pila:* `x` *por valor y* `vector` *por dirección. El resultado se devuelve en el registro r29. Para programar esta subrutina, no es necesario que cree un marco de pila.*

b) Subrutina `DIAGONAL(m, matriz, resultado)` *que determina cuántos elementos distintos de cero contiene la diagonal principal de una matriz de enteros de una palabra, almacenada en memoria por filas. La matriz es cuadrada de orden m, siendo* $0 < m < 256$. *Los parámetros se pasan en la pila: m por valor y los otros por dirección.*

Esta subrutina extrae los elementos de la diagonal principal de la matriz almacenándolos en un vector como variables locales y hace uso de la subrutina `CEROS` *descrita en el apartado anterior. Para su realización se llevará a cabo el tratamiento del marco de pila descrito en clase y se supondrá que están definidas todas las macros que se han explicado en la parte teórica de la asignatura, que la subrutina llamante deja disponibles todos los registros (excepto r1, r30 (SP) y r31 (FP)), que la pila crece hacia direcciones de memoria decrecientes y que el puntero de pila apunta a la última posición ocupada de la cima de la pila.*

SOLUCIÓN

a) La subrutina CEROS recibe en la pila la dirección del vector y el valor que corresponde a su número de elementos. Para recorrer el vector y obtener los elementos que son cero, se utiliza un puntero al vector, un contador de elementos y un contador de ceros que será el propio registro r29 sobre el que debe devolver el resultado.

```
CEROS:      PUSH(r1)              ; salvaguardar dirección de retorno
            ld r3, r30, 4         ; r3 contador de elementos (parámetro x)
            ld r21, r30, 8        ; r21 puntero a vector (parámetro vector)
            addu r29, r0, r0      ; r29 contador de ceros
bucleC:     ld r6, r21, r0        ; cargar elemento
            cmp r5, r6, r0
            bb1 ne, r5, sigue
            addu r29, r29, 1      ; incrementar contador de ceros
sigue:      subu r3, r3, 1        ; decrementar contador de elementos
            cmp r5, r3, r0
            bb1 eq, r5, salir
            addu r21, r21, 4      ; incrementar puntero a vector
            br bucleC
salir:      POP(r1)               ; recuperar dirección de retorno
            jmp(r1)               ; retornar
```

b) La subrutina DIAGONAL recibe en la pila la dirección del resultado, la dirección de la matriz y el orden de la matriz cuadrada almacenada por filas.

Tras crear el marco de pila se leerán los dos primeros parámetros, reservándose tantas palabras en la pila como elementos tiene la diagonal de la matriz y se extraerán los elementos de la diagonal principal para almacenarlos como un vector, en las respectivas posiciones reservadas para variables locales. Para ello se utiliza un puntero a fila, un contador de columna y un puntero al vector.

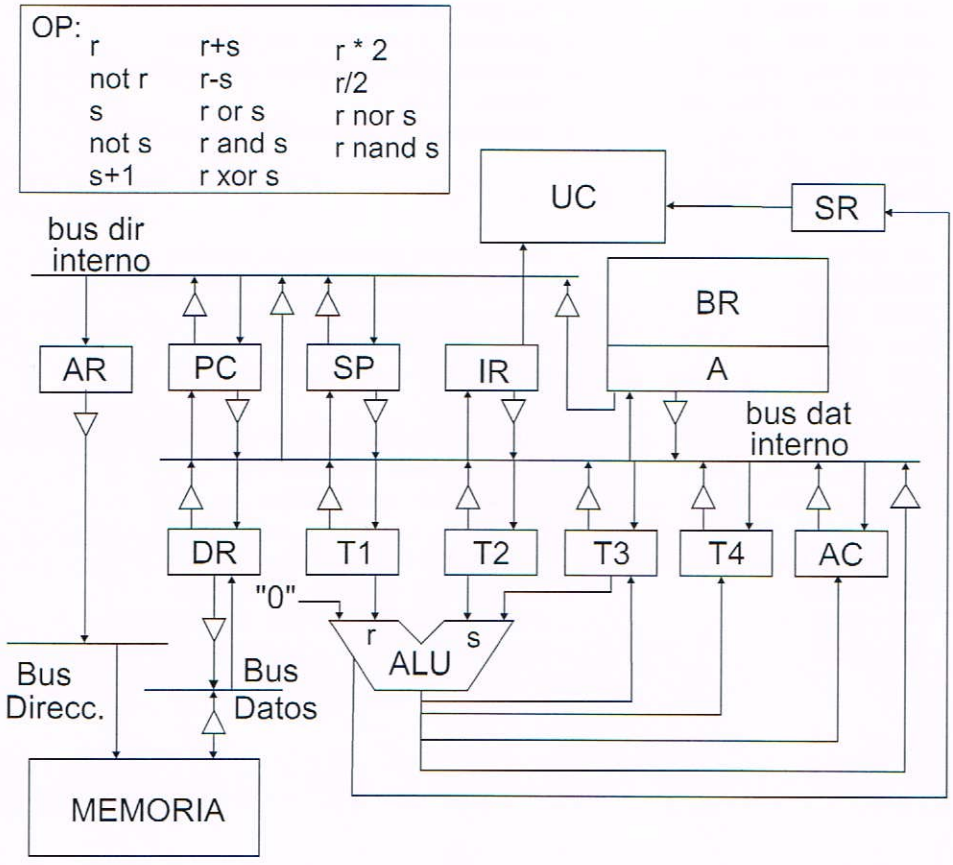
Seguidamente, puesto que esta rutina hace uso de la subrutina CEROS, se pasa en la pila la dirección del vector y su número de elementos. En el retorno, se recuperan los parámetros y se determina el número de elementos distintos de cero. Finalmente, se lee el tercer parámetro para almacenar el resultado y se destruye el marco de pila.

```
DIAGONAL:   PUSH(r1)              ; salvaguardar dirección de retorno
            PUSH (r31)            ; salvaguardar FP del llamante
            or r31, r30, r0       ; crear marco de pila
```

```
                ld r3, r31, 8           ; leer m
                mulu r4, r3, 4          ; tamaño de una fila
                subu r30, r30, r4       ; reservar espacio para el vector
                ld r20, r31, 12         ; cargar dirección de matriz
                or r7, r0, r0           ; r7 contador de columna
                or r21, r30, r0         ; r21 puntero al vector

bucleD:         ld r6, r20, r7          ; cargar elemento
                st r6, r21, r0          ; guardar elemento en vector
                addu r21, r21, 4        ; incrementar puntero al vector
                addu r20, r20, r4       ; nueva fila
                addu r7, r7, 4          ; incrementar contador de columna
                cmp r5, r7, r4
                bb1 ne, r5, bucleD

contar:         or r21, r30, r0         ; recuperar puntero a vector
                PUSH(r21)               ; pasar parámetro por dirección
                PUSH (r3)               ; pasar parámetro por valor
                bsr CEROS               ; llamada a subrutina
                POP (r3)                ; recuperar parámetros
                POP (r21)
                subu r5, r3, r29        ; número de elementos distintos de cero
                ld r22, r31, 16         ; leer tercer parámetro
                st r5, r22, r0          ; almacenar resultado

                or r30, r31, r0         ; destruir variables locales
                POP (r31)               ; recuperar FP del llamante
                POP (r1)                ; recuperar dirección de retorno
                jmp (r1)                ; retornar
```

1 *En la siguiente figura se muestra el esquema de un computador con palabras y direcciones de 64 bits y direccionamiento a nivel de palabra, que incluye dos buses internos al procesador (datos y direcciones), cuatro registros transparentes (T1, T2, T3 y T4) y un registro acumulador (AC). El Banco de registros tiene un puerto de entrada y dos de salida. Los accesos a memoria tienen una duración de 2 ciclos de reloj. El incremento o decremento de cualquier registro, se realiza a través de la ALU, cuyas operaciones se muestran en el recuadro superior.*

Nota: *Por simplificación en la figura, no se han detallado en ella las señales de control.*



a) Considerando que la Unidad de Control es ***cableada****, detalle a nivel RT (transferencia entre registros) las operaciones elementales que se producirán en cada ciclo de reloj, durante la fase de fetch de una instrucción.*

b) Si el computador trabaja a una frecuencia de 400 MHz, determine el tiempo que tarda en ejecutarse la fase de fetch especificada en el apartado anterior.

c) Teniendo en cuenta ahora, que la Unidad de Control es ***microprogramada****, realice (a nivel RT) el microprograma de la fase de ejecución de la instrucción de una palabra:* `XMEM_NC .R2, #desp[.R3 ]`
Esta instrucción realiza el intercambio entre un registro y una posición de memoria, si no está activado el acarreo.

SOLUCIÓN

a)

La fase de fetch puede realizarse en cuatro ciclos, solapando operaciones elementales. En el primero se cargan, simultaneamente, el registro de direcciones de memoria y un registro temporal con el contenido del PC y en el segundo, que corresponde al primer ciclo de acceso a memoria, se actualiza el PC.

```
f1:  T2, AR ← PC
f2:  Acceso a memoria, PC ← T2 + 1
f3:  DR ← M(AR)
f4:  IR ← DR
```

b)

$fr = 400MHz = 4 \cdot 10^8 Hz$

$Tck = 1/400MHz = 2,5 \cdot 10^{-9} s = 2,5ns$

$t_{fetch} = Tck \cdot n^o ciclos = 2,5ns/ciclo \cdot 4ciclos = 10ns$

c)

El microprograma de la fase de ejecución de la instrucción es el siguiente:

```
m1:   Si C microsalto a fetch
m2:   T1 ← BR(R3)          ; obtener la dirección de memoria
m3:   T2 ← IR(desp)
m4:   AR ← T2+T1
m5:   Acceso a memoria
m6:   DR ← M(AR)           ; contenido de memoria a intercambiar
m7:   T4 ← DR
m8:   DR ← BR(R2)
m9:   Acceso a memoria, BR(R2) ← T4    ; cargar contenido de memoria en el registro
m10:  M(AR) ← DR, microsalto a fetch    ; escribir contenido del registro en memoria
```

2 *Un computador cuenta con un formato de representación de números en coma flotante de 16 bits. El bit superior representa el signo del número, los cinco siguientes el exponente y los diez siguientes la mantisa.*

El formato sigue las convenciones del formato estándar IEEE754 en todo excepto en el tamaño, es decir, usa el mismo tipo de representaciones especiales, representación del exponente, bit implícito, situación de la coma, representaciones normalizadas y no normalizadas, así como bits de guarda y modos de redondeo.

a) Determine el rango de representación y la resolución del formato.

b) Represente los siguientes números:

$$A = +129 \quad B = -5,2 \quad C = 5 \cdot 2^{-20} \quad D = -\infty$$

c) Realice paso a paso la suma $A + B$ dejando el resultado en el formato de almacenamiento en memoria.

d) Determine el error absoluto cometido en la operación anterior.

SOLUCIÓN

a) Rango y resolución del formato. Números normalizados.

Exponente: El estándar IEEE754 representa los exponentes en exceso a $2^{n-1} - 1$. En este caso sería exceso a 15. Como se reserva el exponente mínimo (-15) para la representación del cero y los números no normalizados, y el exponente máximo $(+16)$ para la representación del infinito y las indeterminaciones, el rango de exponente para la representación de números queda: $[-14, 15]$.

La mantisas normalizadas en este formato se representan en signo-magnitud, con bit implícito y la coma situada a la derecha del bit implícito:

$$\text{Mantisa: } \pm \begin{cases} 1,0000000000 \rightarrow 1 \\ 1,1111111111 \rightarrow 2 - 2^{-10} \end{cases}$$

El rango para números normalizados es: $\pm [1 \cdot 2^{-14}, (2 - 2^{-10}) \cdot 2^{15}]$

Números no normalizados

Exponente: Siguiendo el estándar IEEE754, aunque los números no normalizados se representan con el exponente todo a ceros (valor -15), todos ellos tienen como exponente el más pequeño de los normalizados, en este caso -14. De este modo hay continuidad en la representación.

Mantisas: $\pm \begin{cases} \texttt{0,0000000000} & \rightarrow & 0 \\ \texttt{0,0000000001} & \rightarrow & 2^{-10} \\ \texttt{0,1111111111} & \rightarrow & 1-2^{-10} \end{cases}$

El rango para números no normalizados es: $\pm\,[2^{-10}\cdot 2^{-14}, (1-2^{-10})\cdot 2^{-14}] \cup 0$

Así pues, el rango total es:

$\pm\,[1\cdot 2^{-14}, (2-2^{-10})\cdot 2^{15}] \cup \pm\,[2^{-10}\cdot 2^{-14}, (1-2^{-10})\cdot 2^{-14}] \cup 0$

La resolución depende del exponente y es: $2^{-10}\cdot 2^{E}$

b) Representación de los números.

$A = \texttt{+129} = \texttt{H'81} = \texttt{+10000001} = \texttt{+1,0000001000}\cdot 2^{7} = \texttt{0 10110 0000001000} = \texttt{H'5808}$

$B = \texttt{-5,2} = \texttt{H'-5,333} = \texttt{-101,00110011} = \texttt{-1,0100110011}\cdot 2^{2} = \texttt{1 10001 0100110011} = \texttt{H'C533}$

$C = \texttt{+5}\cdot 2^{-20} = \texttt{+101}\cdot 2^{-20}$

El número no se puede representar como normalizado porque el exponente se saldría de rango. Se puede representar como normalizado. Buscamos el exponente -14.

$C = \texttt{+0,0001010000}\cdot 2^{-14} = \texttt{0 000000001010000} = \texttt{H'0050}$

$D = -\infty = \texttt{1 11111 0000000000} = \texttt{H'FC00}$

c) Suma $A + B$.

Este formato utiliza dos bits de guarda y un bit retenedor para la suma. Los números a sumar son:

$A = \texttt{+1,0000001000}\cdot 2^{7}$ *y* $B = \texttt{-1,0100110011}\cdot 2^{2}$

Se restan los exponentes: $E_A - E_B = 7 - 2 = 5$. Hay que desplazar la mantisa de B cinco lugares a la derecha. Así, teniendo en cuenta los dos bits de guarda y el bit retenedor queda:

$B = \texttt{-0,0000101001101}\cdot 2^{7}$

M_A	1,0000001000 000
M_B	− 0,0000101001 101
	0,1111011110 011 $\cdot 2^{7}$
Normalizacion	1,1110111100 110 $\cdot 2^{6}$
Redondeo	0,0000000000 100
	1,1110111101 010 $\cdot 2^{6}$

$A + B = \texttt{+1,1110111101}\cdot 2^{6} = \texttt{0 10101 1110111101} = \texttt{H'57BD}$

d) Error.

Sumando los valores decimales de A y B queda:

$A + B = \texttt{129 - 5,2} = \texttt{123,8}$

El resultado de la suma en coma flotante ha sido:

$A + B = \texttt{+1,1110111101}\cdot 2^{6} = \texttt{+1111011,1101} = \texttt{123,8125}$

El error aboluto:

$e_A = ||\texttt{123,8} - \texttt{123,8125}|| = \texttt{0,0125}$

ESTRUCTURA DE COMPUTADORES
PRIMER PARCIAL (26 de Abril de 2013)

Apellidos, Nombre..... Nº de Matrícula.....

Responda en esta misma hoja, utilizando únicamente el espacio asignado para cada pregunta.

1 (1,5 puntos) *Indique las fases de ejecución de la instrucción de una palabra* `PUSH #4[.R4]`

```
1) Fetch
. AR <- PC
. DR <- M, PC++
. RI <- DR
2) Decodificación y lect ops
3) Ejecución y esc. resultados
. AR <- RI(d) + R4
. DR <- M, SP--
. AR <- SP
. M <- DR
```

2 (2,5 puntos) *Sea un computador de 32 bits cuyos únicos modos de direccionamiento son: inmediato, directo a registro e indirecto a registro y todas sus instrucciones ocupan una palabra. El modelo de ejecución es registro-registro y la memoria es direccionable a nivel de byte. Indique las secuencias de instrucciones equivalentes a las que se expresan a continuación en ensamblador del IEEE, utilizando como registros temporales .RT1, .RT2, .RT3, .RT4 y .RT5:*

CALL [/1000]

```
LD .RT1, #1000
LD .RT1, [.RT1]
CALL [.RT1]
```

MOVE /5000, #7[.R2--]

```
LD .RT1, #5000
LD .RT2, #7
ADD .RT2, .R2
MOVE [.RT1], [.RT2]
SUB .R2, #4
```

ADDV #4, /1000, /2000 (Suma dos vectores de 4 elementos almacenados en las posiciones 1000 y 2000, almacenando el resultado en el primero de ellos)

```
     LD .RT1, #4
     LD .RT2, #1000
     LD .RT3, #2000
BUC: LD .RT4,[.RT2]
     LD .RT5,[.RT3]
     ADD .RT4,.RT5
     ST .RT4,[.RT2]
     ADD .RT3,#4
     ADD .RT2,#4
     SUB .RT1,#1
     BNZ BUC
```

3 (6 puntos) *Se desea realizar una operación de suma de dos submatrices pertenecientes a una misma matriz de enteros. La función* `suma_submatriz` *recibe los siguientes parámetros en la pila:*

- `matriz`*: Es una matriz de enteros. Se pasa por dirección.*
- `filas`, `columnas`*: Son dos valores enteros que contienen las dimensiones de la matriz: número de filas y columnas. Se pasan por valor.*
- `filam1`, `columnam1`*: Son dos valores enteros que contienen el número de fila y columna del primer elemento de la primera submatriz en la matriz original. Las filas y las columnas comienzan a numerarse en la posición 0. Se pasan por valor.*
- `filam2`, `columnam2`*: Son dos valores enteros que contienen el número de fila y columna del primer elemento de la segunda submatriz en la matriz original. Las filas y las columnas comienzan a numerarse en la posición 0. Se pasan por valor.*
- `nfilas`, `ncolumnas`*: Son dos valores enteros que contienen el número de filas y columnas de las dos submatrices. Se pasan por valor.*
- `resultado`*: Es una matriz de enteros de* `nfilas` *filas y* `ncolumnas` *columnas. Contiene la submatriz resultado de la suma. Se pasa por dirección.*

Programe las siguientes subrutinas:

a) La función indicada a continuación copia la submatriz indicada por los cuatro primeros parámetros en la dirección `result`*:*

```
extraer_submatriz(filam,columnam,filasm,columnasm,matriz,filas_matriz,columnas_matriz,result)
```

A modo de ejemplo la llamada `extraer_submatriz(0, 0, 2, 3, matriz, 3, 6, result)` *copiaría la primera submatriz del ejemplo que comienza en el elemento (0,0) y tiene dos filas y tres columnas de la matriz original (*`matriz`*) que tiene tres filas y seis columnas. Todos los parámetros se pasan por valor, excepto la matriz y el resultado que se pasan por dirección.*

b) La función `suma(m1,m2,nfilas,ncolumnas)` *suma dos matrices (*`m1` *y* `m2`*) almacenadas por filas almacenando el resultado en* `m1`*. Las dos matrices se pasan por dirección y las dimensiones de la matriz por valor.*

c) La función `suma_submatriz` *descrita en los párrafos anteriores. Para facilitar la realización del apartado se supondrá que los parámetros son correctos y se seguirán los siguientes pasos:*

1. *Se copia la primera submatriz en el parámetro* `resultado` *invocando a* `extraer_submatriz`
2. *Se copia la segunda submatriz en una variable local creada previamente en el marco de pila de la subrutina invocando a* `extraer_submatriz`
3. *Se suman las submatrices, invocando a* `suma`*, almacenando el resultado en* `resultado`*.*

La matriz original contiene las dos submatrices que están resaltadas en negrita. La primera comienza en el elemento (0,0) y la segunda en el elemento (1.3) y son de dos filas y tres columnas. El resultado es la suma de estas dos submatrices.

$$Matriz = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 7 & 8 & 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 & 17 & 18 \end{pmatrix}$$

$$Resultado = \begin{pmatrix} 11 & 13 & 15 \\ 23 & 25 & 27 \end{pmatrix}$$

SOLUCIÓN

a) La subrutina ejecuta dos bucles anidados para realizar la copia de la submatriz. Inicialmente se calcula la dirección del primer elemento y se realiza la copia de la primera fila. Para pasar a la siguiente fila se incrementa la dirección del primer elemento de la fila en el tamaño de la fila de la matriz que se ha calculado en r5.

```
extrae_submatriz:
        ld r20,r30,28   ; Dir. Resultado
        ld r2,r30,0     ; fila inicio submatriz
        ld r3,r30,4     ; columna inicio submatriz
        ld r4,r30,20    ; filas matriz original
        ld r5,r30,24    ; columnas matriz original
        mulu r5,r5,4    ; tamaño de fila de matriz original
        ld r21,r30,16   ; Dir. matriz
        ld r9,r30,12    ; Número columnas submatriz
        ld r10,r30,8    ; Número filas submatriz
        or r11,r9,r9    ; Copia de número columnas submatriz
        mulu r2,r2,r5
        mulu r8,r3,4
        addu r2,r8,r2
        addu r21,r21,r2 ; Dirección del primer elemento de submatriz
bucle_e: or r22,r21,r21
bucle_i: ld r12,r22,r0
        st r12,r20,r0   ; Copia elemento en resultado
        addu r22,r22,4
        addu r20,r20,4
        subu r11,r11,1
        cmp r12,r11,r0
        bbl ne,r12,bucle_i
        or r11,r9,r9    ; Si no hay más filas -> fin
        subu r10,r10,1
        cmp r12,r10,r0
        bbl eq,r12,fin
        addu r21,r21,r5 ; Cálculo del primer elemento de la siguiente fila
        br bucle_e
fin:    jmp(r1)
```

b) La suma de dos matrices se realiza por el procedimiento simplificado. Puesto que las matrices están almacenadas por filas, se suman las dos como si fueran dos vectores de `nfilas` x `ncolumnas`.

```
suma: ld r2,r30,12      ; Num. columnas
      ld r3,r30,8       ; Num. filas
      mulu r2,r2,r3     ; Tamaño matriz en palabras
      ld r20,r30,r0     ; Puntero a m1
      ld r21,r30,4      ; Puntero a m2
      or r4,r0,r0       ; Desplazamiento del elemento evaluado
                        ; frente al comienzo de las matrices
bucle: ld r3,r21,r4     ; Elemento de m2
      ld r5,r20,r4      ; Elemento de m1
      add r3,r3,r5      ; Suma
      st r3,r20,r4      ; Almacenamiento en m1
      addu r4,r4,4      ; Incrementa el desplazamiento
      subu r2,r2,1      ; Decrementa contador elementos
      cmp r3,r2,r0
      bbl ne,r3,bucle
      jmp(r1)
```

c) La suma de las dos submatrices se realiza siguendo la secuencia inidicada en el enunciado. Los comentarios del código indica qué se ejecuta en cada fragmento del programa.

```
suma_submatriz:
      PUSH(r1)
      PUSH(r31)
      or r31,r30,r30
      ld r2,r31,36      ; filas submatriz
      ld r3,r31,40      ; columnas submatriz
```

```
mulu r2,r2,r3      ; Se genera espacio en la pila
mulu r2,r2,4       ; para una submatriz (variable local)
subu r30,r30,r2

ld r20,r31,44      ; Dir resultado
PUSH(r20)
ld r2,r31,16       ; Columnas Matriz
ld r3,r31,12       ; Filas Matriz
PUSH(r2)
PUSH(r3)
ld r20,r31,8       ; Dir. matriz
PUSH(r20)
ld r2,r31,40       ; Nº columnas de la submatriz
ld r3,r31,36       ; Nº filas de la submatriz
PUSH(r2)
PUSH(r3)
ld r2,r31,24       ; Columna de inicio de la submatriz
ld r3,r31,20       ; Fila de inicio de la submatriz
PUSH(r2)
PUSH(r3)
bsr extrae_submatriz
addu r30,r30,32
or r20,r30,r30     ; Dir. submatriz en var local
PUSH(r20)          ; Se realizan las mismas acciones que
ld r2,r31,16       ; en la llamada anterior, pero para la
ld r3,r31,12       ; segunda submatriz, es decir,
PUSH(r2)           ; la que se almacena en la var. local
PUSH(r3)
ld r20,r31,8
PUSH(r20)
ld r2,r31,40
ld r3,r31,36
PUSH(r2)
PUSH(r3)
ld r2,r31,32
ld r3,r31,28
PUSH(r2)
PUSH(r3)
bsr extrae_submatriz
addu r30,r30,32
or r20,r30,r0      ; Dir. submatriz en var local
ld r2,r31,40       ; Nº columnas
PUSH(r2)
ld r2,r31,36       ; Nº filas
PUSH(r2)
PUSH(r20)
ld r20,r31,44
PUSH(r20)          ; Dir. resultado
bsr suma
or r30,r31,r31
POP(r31)
POP(r1)
jmp(r1)
```

ESTRUCTURA DE COMPUTADORES (Grado I.I.)

RECUPERACIÓN DEL PRIMER PARCIAL (18 de enero de 2013)

---

1 (2 puntos) *Indique, justificando su respuesta, si las siguientes afirmaciones son verdaderas o falsas:*

a) La Entrada/Salida por interrupciones es la que proporciona mejor rendimiento en la transferencia de un bloque de datos a un periférico.

Falso. La entrada salida mediante acceso directo a memoria (DMA) es la que proporciona mejor rendimiento en dispositivos de bloque al transmitir la información el dispositivo sin necesidad de intervención de la CPU.

a) La Unidad de Control no necesita como entrada el registro contador de programa (PC).

Verdadero. Para la unicdad de control el PC es un registro más. Los únicos que son entradas directas son el registro de estado y el registro de instrucción.

a) El registro de estado es un registro transparente al usuario, ya que este no tiene por qué utilizarlo en las instrucciones máquina.

Falso. El registro de estado aparece en la especificación del juego de instrucciones del computador (por ejemplo en las instrucciones de salto condicional) y por tanto no es un registro transparente.

2 (8 puntos) *La tabla no ordenada* `ALUMNOS` *contiene la información de los alumnos que cursan una asignatura. El tamaño de la entrada de la tabla es de 120 bytes, su primer campo es un entero de 4 bytes que contiene el identificador del alumno y el último campo es otro entero del mismo tamaño que debe contener la nota media de un examen y que inicialmente contiene el valor* `0xffffffff`*. El fin de la tabla está indicado con el valor* `0xffffffff` *en el campo de identificador del alumno.*

La tabla `EXAMEN` *contiene las calificaciones de tres problemas de un examen. Su primer campo es un entero de 4 bytes que contiene el identificador del alumno y los otros tres son tres enteros de una palabra que contienen dichas calificaciones. El marcador de fin de tabla sigue el mismo criterio que en la tabla anterior.*

La tabla `INCIDENCIAS` *contiene las incidencias de un examen. Consta de dos campos enteros: el identificador del alumno y el tipo de incidencia (0 si el alumno no se ha presentado al examen y 1 si se ha presentado y no está en la tabla* `ALUMNOS`*). El marcador de fin de tabla sigue el mismo criterio que en la tabla anterior.*

Se desea actualizar el último campo de la tabla `ALUMNOS` *con los registros de la tabla* `EXAMEN` *y generar la tabla* `INCIDENCIAS` *que contenga las incidencias de las calificaciones. Para ello debe realizar:*

a) la función `dir_entrada=BUSCAR (id_alumno, Tabla)` *que busca la entrada que contiene al alumno identificado por* `id_alumno` *(entero pasado por valor) en la tabla que se pasa en el segundo parámetro (por dirección). Esta subrutina devuelve la dirección de comienzo de la entrada de la tabla que contiene al alumno indicado en* `id_alumno`*. Si el alumno no se ha encontrado, devuelve* `0xffffffff`*.*

b) la subrutina `ACTUALIZAR(Tabla_Alumnos, Tabla_Examen, Tabla_Incidencias)` *recibe tres tablas pasadas por dirección. La subrutina realiza las siguientes acciones:*

- *Actualiza el último campo de la tabla* `Tabla_Alumnos` *para todos los alumnos que aparezcan en* `Tabla_Examen` *con la nota media de las tres calificaciones.*
- *Los alumnos que están contenidos en* `Tabla_Examen` *y no están contenidos en* `Tabla_Alumnos` *deben aparecer en la tabla* `Tabla_Incidencias` *con el tipo de incidencia 1.*
- *Los alumnos que no aparecen en* `Tabla_Examen` *y sí aparecen en* `Tabla_Alumnos` *deben aparecer en la tabla* `Tabla_Incidencias` *con el tipo de incidencia 0.*

Se supondrá que están definidas todas las macros que se han explicado en la parte teórica de la asignatura, que la subrutina llamante deja disponibles todos los registros excepto r1, r30 (SP) y r31 (FP); que la pila crece hacia direcciones de memoria decrecientes y el puntero de pila apunta a la última posición ocupada (de la misma forma que se ha utilizado en el proyecto) y que todos los parámetros de las subrutinas anteriores se pasan en la pila. A modo de ejemplo, a continuación se muestra una situación inicial de las tablas `ALUMNOS` *y* `EXAMEN` *y el resultado en las tablas* `ALUMNOS` *e* `INCIDENCIAS`*.*

ALUMNOS

id_alumno	*campos*	*media*
1	...	*-1*
4	...	*-1*
-1	*??*	*??*

EXAMEN

id_alumno	*P1*	*P2*	*P3*
4	*1*	*2*	*3*
5	*4*	*5*	*6*
-1	*??*	*??*	*??*

Las tablas resultado son:

ALUMNOS

id_alumno	*campos*	*media*
1	...	*-1*
4	...	*2*
-1	*??*	*??*

INCIDENCIAS

id_alumno	*Tipo*
5	*1*
1	*0*
-1	*??*

SOLUCIÓN

a) La subrutina buscar carga el elemento que se pasa en la pila y va recorriendo la tabla que se pasa como parámetro. El registro r29 se inicializa con el valor 0xffffffff y solo si se encuentra se carga con el puntero que recorre la tabla (r20).

```
buscar: PUSH(r1)
        ld r20,r30,8
        ld r3,r30,4
        sub r29,r0,1
bucle_b:  ld r4,r20,r0
        cmp r5,r4,-1
        bb1 eq, r5, fin
        cmp r5,r3,r4
        bb1 eq, r5, encont
        addu r20,r20,120
        br bucle_b
encont: or r29,r20,r20
fin:    POP(r1)
        jmp(r1)
```

b) La solución de la rutina actualizar crea el marco de pila para contener dos variables locales. El puntero a la entrada en curso de la tabla de incidencias ocupa el desplazamiento -4 al puntero de marco r31 y se inicializa con la dirección de comienzo de la tabla de incidencias. El puntero a la entrada en curso de la tabla de exámenes ocupa el desplazamiento -8 y se inicializa con la dirección de comienzo de la tabla de exámenes.

Se recorre la tabla de exámenes. Para cada entrada se busca si existe en la tabla de alumnos. Si existe, se calcula la media y se almacena en la tabla de alumnos. Si no, se da de alta en la tabla de incidencias con el valor de incidencia 1. Una vez que se ha finalizado el recorrido de la tabla de exámenes se recorre la tabla de alumnos. Todos aquellos cuya media contiene el valor -1 no se han presentado al examen y por tanto se dan de alta en la tabla de incidencias con el valor 0. Cuando se ha finalizado el recorrido de la tabla de alumnos se almacena la marca de fin de tabla en la tabla de incidencias.

```
actualizar: PUSH(r1)
            PUSH(r31)
            or r31,r30,r30
            subu r30,r30,8
            ld r20,r31,16       ; tabla incidencias
            sub r3,r0,1
            st r3,r20,r0
            ld r21,r31,12       ; tabla examen
bucle:      ld r3,r21,r0        ; id_alumno de tabla_examen
            st r20,r31,-4       ; puntero a tab_incidencias (var local)
            st r21,r31,-8       ; puntero a tab_examen (var local)
            cmp r5,r3,-1
```

```
            bb1 eq, r5, fin_buc
            ld r20,r31,8        ; tabla alumnos
            PUSH(r20)
            PUSH(r3)
            bsr buscar
            cmp r5,r29,-1
            bb1 eq, r5, no_esta
            ld r21,r31,-8       ; El alumno está
            ld r3,r21,4         ; Se accede a las notas, se calcula la media
            ld r4,r21,8         ; y se almacena en tabla_alumnos
            add r3,r3,r4
            ld r4,r21,12
            add r3,r3,r4
            divu r3,r3,3
            st r3,r29,116
            addu r21,r21,16     ; Se avanza al siguiente
            ld r20,r31,-4
            br bucle
no_esta:    ld r20,r31,-4       ; El alumno no está
            ld r21,r31,-8       ; Se almacena el alumno con la incidencia
            ld r3,r21,0         ; con valor 1 en tabla_incidencias
            st r3,r20,0
            add r3,r0,1
            st r3,r20,4
            sub r3,r0,1
            st r3,r20,8         ; Se almacena la marca de fin de tabla
            addu r21,r21,16     ; Se incrementan punteros
            addu r20,r20,8
            br bucle
fin_buc:    ld r20,r31,8        ; tabla_alumnos
            ld r21,r31,-4       ; tabla_incidencias
buc2:       ld r3,r20,r0        ; Se buscan todas las entradas que tengan media -1
            cmp r5,r3,-1
            bb1 eq, r5, final
            ld r2,r20,116       ; Estas entradas se añaden a tabla_incidencias con
            cmp r5,r2,-1        ; el valor de incidencia a 0
            addu r20,r20,120
            bb1 ne, r5, buc2
            st r3,r21,r0
            st r0,r21,4
            addu r21,r21,8
            br buc2
final:      sub r3,r0,1         ; Se almacena la marca de fin de tabla en la tabla
            st r3,r21,r0        ; de incidencias
            or r30,r31,r31
            POP(r31)
            POP(r1)
            jmp(r1)
```

NOTAS: 29 de enero de 2013
REVISIÓN: 30 de enero de 2013

DURACIÓN: 1 hora 30 minutos
PUNTUACIÓN: Especificada en cada problema

1 (5 puntos) *Sea la estructura del procesador de la figura 1 con unidad de control microprogramada. La memoria de este computador es asíncrona y activa una señal RDY cuando ha acabado la operación que se le ha solicitado. Se quiere dotar a este procesador de la instrucción de* ***tres palabras*** `ADD /dir1, /dir2`. *Esta instrucción suma dos palabras que están en memoria en las direcciones dir1 y dir2 respectivamente y deja el resultado en la primera dirección.*

a) Indique qué cambios habría que hacer en la estructura de este procesador para poder ejecutar esta instrucción. Justifique la respuesta.

b) Teniendo en cuenta su respuesta del apartado anterior, microprograme a nivel RT la instrucción `ADD /dir1, /dir2`, *incluyendo la microsubrutina de fetch, y solapando operaciones elementales siempre que sea posible.*

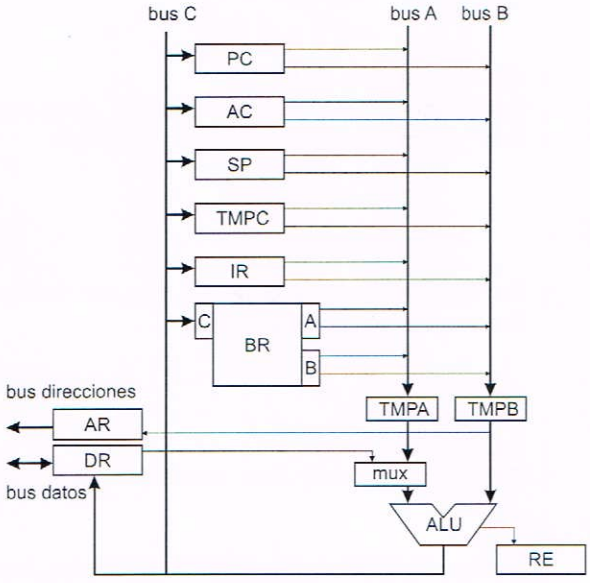


Figura 1. Estructura de la CPU

SOLUCIÓN

a) Para ejecutar esta instrucción haría falta al menos algún registro temporal más. Podría considerarse que pudiera incluirse en el banco de registros, o bien añadir otro registro TMPD conectado igual que está TMPC. Algunas conexiones nuevas podrían mejorar el rendimiento, como conectar los buses A y B directamente con AR, para no tener que guardar las direcciones siempre en TMPB antes de operar con la memoria, aunque éstas no son estrictamente necesarias. En el apartado siguiente sólo se ha tenido en cuenta como modificación la inclusión del registro temporal TMPD.

b) El microprograma de fetch es el siguiente:

```
f1:  PC → TMPB
f2:  TMPB → AR, TMPB+1 →PC
f3:  M(AR) → DR, si RDY ir a f3
f4:  DR+0 → IR, ir a C0
```

El microprograma de la ejecución de la instrucción es el siguiente:

```
a1:   PC → TMPB
a2:   TMPB → AR, TMPB+1 →PC
a3:   M(AR) → DR, si RDY ir a a3
a4:   DR+0 → TMPC                    leer dir1
a5:   PC → TMPB
a6:   TMPB → AR, TMPB+1 →PC
a7:   M(AR) → DR, si RDY ir a a7
a8:   DR+0 → TMPD                    leer dir2
a9:   TMPD → TMPB
a10:  TMPB → AR
a11:  M(AR) → DR, si RDY ir a a11
a12:  DR+0 → TMPD                    leer dato2
a13:  TMPC → TMPB
a14:  TMPB → AR
a15:  M(AR) → DR, si RDY ir a a15    leer dato1
a16:  DR+TMPD → DR                   sumar datos
a17:  DR → M(AR), si RDY ir a a17    escritura en dir1
a18:  ir a fetch (f1)
```

2 *Un computador cuenta con los siguientes formatos de representación:*

Formato 1. *Coma fija: 16 bits en complemento a uno con la coma entre los bits 8 y 7 (8 bits de parte entera y otros 8 de parte fraccionaria), por ejemplo 11010110,11101011.*

Formato 2. *Coma flotante: 16 bits, el bit superior es el signo, los siete siguientes el exponente, representado en exceso a 64 y los 8 siguientes la magnitud de la mantisa, que está representada en signo magnitud, con bit implícito y la coma a la derecha de éste.*

1. *Determine el rango y la resolución de ambos formatos.*
2. *Dados los números decimales A = +16,8 y B = -0,6 represéntelos en ambos formatos.*
3. *Realice paso a paso la operación A+B en ambos formatos, dejando el resultado en el formato de partida. En el caso de coma flotante utilice dos bits de guarda y un bit retenedor, así como redondeo al más próximo.*
4. *Determine el error absoluto que se ha cometido en las operaciones anteriores.*

SOLUCIÓN

a) Rango y resolución de los formatos.

Formato 1.

El rango de una representación en complemento a uno es simétrico. Así:

$$\pm \begin{cases} 00000000,00000000 & \rightarrow & 0 \\ 01111111,11111111 & \rightarrow & 2^7 - 2^{-8} \end{cases}$$

$$Rango = \pm\,[0, 2^7 - 2^{-8}] = [-(2^7 - 2^{-8}), 2^7 - 2^{-8}]$$

$$Resolucion = 2^{-8}$$

Formato 2.

Exponente: $[-64, 63]$. Como hay que reservar el exponente -16 para la representación del cero, el rango del exponente queda: $[-63, 63]$.

$$\text{Mantisa: } \pm \begin{cases} 1,00000000 & \rightarrow & 1 \\ 1,11111111 & \rightarrow & 2 - 2^{-8} \end{cases}$$

$$Rango = \pm\,[1 \cdot 2^{-63}, (2 - 2^{-8}) \cdot 2^{63}] \cup 0$$

$$Resolucion = 2^{-8} \cdot 2^{E}$$

b) **Representación de los númetos**

$A = +16,8 = H'+10,CC = +10000,11001100$

$B = -0,6 = H'-,999 = -,100110011001$

Formato 1.

$A = 00010000,11001100 = H'10CC$

$B = -00000000,10011001 = 11111111,01100110 = H'FF66$

Formato 2.

$A = +1,00001100 \cdot 2^4 = 0\ \ 1000100\ \ 00001100 = H'440C$

$B = +1,00110011 \cdot 2^{-1} = 1\ \ 0111111\ \ 00110011 = H'BF33$

c) **Suma $A + B$.**

Formato 1.

A	00010000, 11001100
$+B$	11111111, 01100110
$A + B$	1 00010000, 00110010
+carry	1
$A + B$	00010000, 00110011

$A + B = 00010000,00110011 = H'1033$

Formato 2.

Los números a sumar son:

$A = +1,00001100 \cdot 2^4 \qquad B = +1,00110011 \cdot 2^{-1}$

La diferecia de exponentes: $E_A - E_B = 4 - (-1) = 5$ nos dice que hay que desplazar la mantisa de B cinco lugares a la derecha. Si partimos de:

$B = +1,00110011 \cdot 2^{-1}$

Al desplazar cinco lugares (teniendo en cuenta el bit retenedor) queda:

$B = +0,00001001\ 10\ 1 \cdot 2^4$

Se realiza la suma de mantisas (resta):

M_A	$+\ 1,00001100\ 00\ 0$	
$+M_B$	$-\ 0,00001001\ 10\ 1$	
$A + B$	$+\ 1,00000010\ 01\ 1 \cdot 2^4$	*Normalizado*
	$+\ 0,00000000\ 10\ 0$	*Redondeo*
$A + B$	$+\ 1,00000010\ 11\ 1 \cdot 2^4$	

$A + B = +1,00000010 \cdot 2^4 = 0\ \ 1000100\ \ 00000010 = H'4402$

d) **Error.**

Sumando los valores decimales de A y B queda:

$A + B = 16,8 - 0,6 = 16,2$

Formato 1.

El resultado de la suma en coma fija ha sido:

$A + B = 00010000,00110011 = 16,1992$

El error aboluto:

$e_A = ||16,2 - 16,1992|| = 0,0008$

Formato 2.

El resultado de la suma en coma flotante ha sido:

$A + B = +1,00000010 \cdot 2^4 = +10000,0010 = 16,125$

El error aboluto:

$e_A = ||16,2 - 16,125|| = 0,075$

NOTAS: 29 de enero de 2013
REVISIÓN: 30 de enero de 2013

DURACIÓN: 1 hora 30 minutos
PUNTUACIÓN: Especificada en cada problema

4/

1 *Sea un teléfono móvil cuya pantalla gráfica tiene una resolución de* 480×640 *píxeles y una gama de 65.536 colores. La memoria de pantalla de su tarjeta gráfica accede a palabras de 32 bits con un tiempo de acceso de 100 ns.*

*a) (**1 punto**) Calcule cuál es la máxima frecuencia de refresco a la que puede funcionar la pantalla.*

En el teléfono móvil se ejecuta una aplicación que captura el contenido de la pantalla y lo envía por su conector miniusb usando una transmisión en serie asíncrona de 921.600 bits por segundo, con 8 bits de dato y bit de paridad impar.

*b) (**1 punto**) Si la transmisión de la pantalla emplea 8 segundos, calcule cuántos bits de stop se usan en la transmisión en serie.*

SOLUCIÓN

a) Profundidad de color:

$\log_2 65.536 = 16$ bits $= 2$ bytes

Tamaño de la pantalla:

480×640 pixeles $\times$ 2 bytes/pixel $=$ 614.400 bytes

Ancho de banda de la memoria de pantalla:

$\frac{32/8 \text{ bytes}}{100 \times 10^{-9} \text{ s}} = 40.000.000$ bytes/s

Frecuencia de refresco máxima:

$\frac{40.000.000 \text{ bytes/s}}{614.400 \text{ bytes}} = 65{,}1$ Hz

b) Bits transmitidos en 8 segundos:

8 s $\times$ 921.600 bits/s $=$ 7.372.800 bits

Bits transmitidos por cada byte:

$\frac{7.372.800 \text{ bits}}{614.400 \text{ bytes}} = 12$ bits/byte

Descontando el bit de start, los 8 bits de dato y el bit de paridad, quedan dos bits correspondientes a los bits de stop.

2 *Sea un computador al que están conectados una unidad de cinta magnética y una unidad de disco duro. En el instante t=0 la unidad de cinta se encuentra detenida y las cabezas de grabación de la unidad de disco están situadas al comienzo del sector 120 del cilindro 3.000.*

Características de la unidad de cinta:

- *Tiempo de arranque y de parada: 2,5 ms.*
- *Velocidad de transferencia: 12 Mbytes/s (*12×10^6 *bytes/s).*
- *Claros IRG de 0,4 cm de longitud.*
- *Densidad de grabación lineal: 6.464 bits/mm.*
- *Bloques de 16.384 bytes.*

Características de la unidad de disco duro:

- *Capacidad Bruta: 691.200.000.000 bytes.*
- *18 superficies y 20.000 cilindros.*
- *Sectores de 8.192 bytes de información neta con 1.408 bytes adicionales para direccionamiento, control de errores, gaps, etc.*
- *Velocidad de rotación: 3.750 rpm.*
- *Densidad de grabación lineal máxima: 64.000 bytes/cm.*
- *Tiempo que emplea en mover la cabeza de una pista a otra consecutiva: 0,02 ms.*
- *Tiempo de estabilización de las cabezas: 2 ms.*

A partir del instante t=0, se arranca la cinta, se leen dos bloques consecutivos, se detiene la cinta y se escriben los datos leídos en el disco duro.

*a) (**1 punto**) Calcule en qué instante termina de operar la unidad de cinta.*

*b) (**2 puntos**) Calcule el número de sectores por pista y la velocidad de transferencia de la unidad de disco.*

*c) (**5 puntos**) Si los datos leídos de la cinta se escriben en los sectores 7.203.320, 7.203.321, 5.943.599 y 5.943.600 de la unidad de disco, calcule el instante en el que finaliza la escritura de los datos.*

SOLUCIÓN

a) Tiempo empleado por la cinta:

$2{,}5 \text{ ms} + \frac{16.384+0{,}4\times10\times6.464/8+16.384}{12\times10^6} \text{ s} + 2{,}5 \text{ ms} = 8 \text{ ms}$

b) Número de sectores por pista:

$$\frac{691.200.000.000 \text{ bytes}}{(8.192+1.408) \text{ bytes/sector}\times20.000\times18 \text{ pistas}} = 200 \text{ sectores/pista}$$

Velocidad de transferencia:

$\frac{691.200.000.000 \text{ bytes}}{20.000\times18 \text{ pistas}} = 1.920.000$ bytes/pista

$\frac{3.750rpm}{60s/m} = 62{,}5\ rps$

$1.920.000 \text{ bytes/pista} \times 62{,}5 \text{ pistas/s} = 120 \cdot 10^6 \text{ bytes/s}$

c) Tiempo de giro: 1/62,5 rps = 0,016 s = 16 ms. Tiempo de avance de un sector: 16/200 = 0,08 ms.

Coordenadas geométricas (CHS) de los sectores:

7.203.320/(18 · 200) = 2.000 (resto 57.720)

57.720/200 = 16; resto 120

7.203.320 → (2.000, 16, 120)

7.203.321 → (2.000, 16, 121)

Procediendo de igual forma para los demás:

5.943.599 → (1650, 17, 199)

5.943.600 → (1651, 0, 0)

Sector 7.203.320:

Tbúsqueda = (3.000 - 2.000)0,02 + 2 = 22 ms (Teniendo en cuenta los 8 ms que emplea la cinta, 22 + 8 = 30 ms, 1 vuelta completa más 14 ms durante los cuales avanza 14/0,08 = 175 sectores, quedando las cabezas delante del sector 95 ([175 + 120] mod 200).

Tlatencia = (120 - 95)0,08 = 2 ms

Lectura = 0,08 ms

Total: 22 + 2 + 0,08 = 24,08 ms

Sector 7.203.321:

Tacceso = 0

Lectura = 0,08 ms

Total : 0,08 = 0,08 ms

Sector 5.943.599:

Tbúsqueda = (2.000 - 1650)0,02 + 2 = 9 ms (Avanza 9/0,08=112,5 sectores quedando las cabezas en el medio del sector 34,5 (122+112,5 =234,5)).

Tlatencia = (199 - 34,5)0,08 = 13,16 ms.

Lectura = 0,08 ms

Total : 9 + 13,16 + 0,08 = 22,24 ms

Sector 5.943.599:

Para acceder a este sector es preciso mover las cabezas al siguiente cilindro (1651) empleando 2,02 ms y esperar a que el disco complete una vuelta (16-2,02 = 13,98 ms), quedando nuevamente sobre el sector 0. En total, el tiempo de acceso son 16 ms. El tiempo de escritura de este último sector son 16,08 ms (0,08 ms + 16 ms correspondientes al cambio de cilindro).

Total = 16 + 0,08 = 16,08 ms

Tiempo total empleado en la lectura y escritura de los datos:

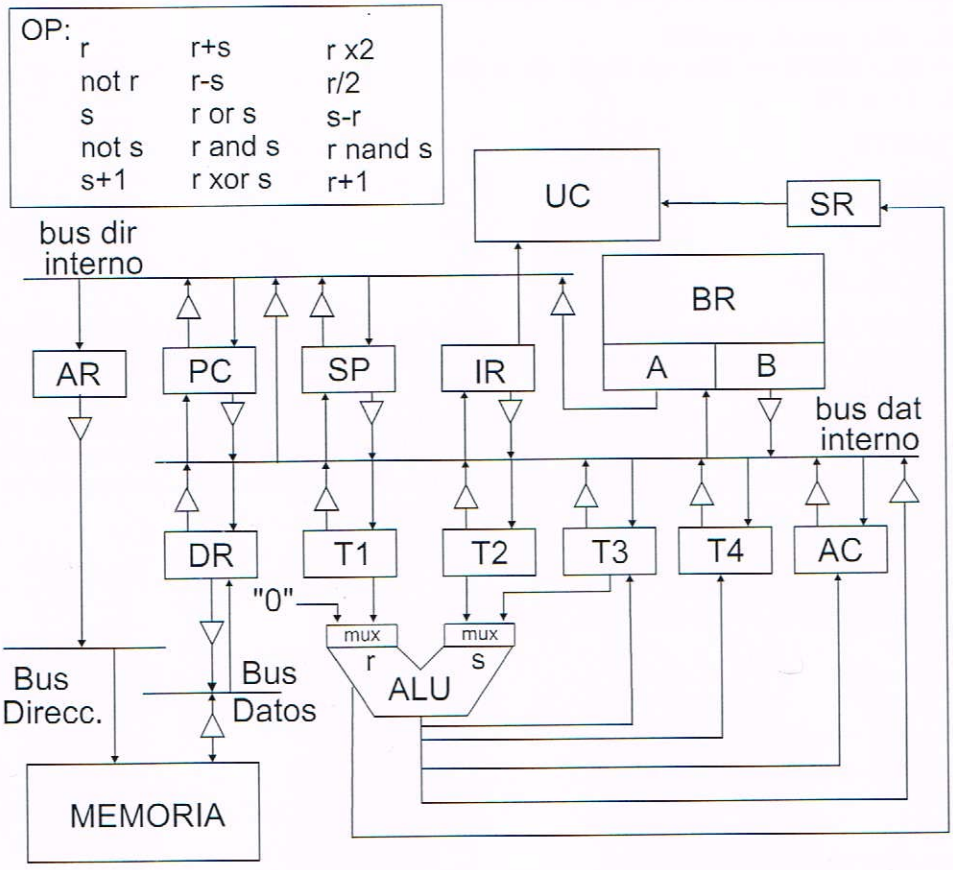
8 + 24,08 +0,08 + 22,24 + 16,08 = 70,48 ms

ESTRUCTURA DE COMPUTADORES (Grado II)
SEGUNDO PARCIAL (24 de mayo de 2013)

---

1 ***(5 puntos)*** *En la siguiente figura se muestra el esquema de un computador con palabras y direcciones de 64 bits y direccionamiento a nivel de byte, que incluye dos buses internos al procesador (datos y direcciones), cuatro registros transparentes (T1, T2, T3 y T4) y un registro acumulador (AC). El Banco de registros tiene un puerto de entrada y dos de salida. La memoria es asíncrona, genera una señal Wait mientras no haya acabado la operación. El incremento o decremento de cualquier registro, se realiza a través de la ALU, cuyas operaciones se muestran en el recuadro superior. Su unidad de control es* ***microprogramada****.*

Nota*: Por simplificación en la figura, no se han detallado en ella las señales de control.*



a) Determine el mínimo tiempo de ciclo de reloj suponiendo que se consideran los siguientes tiempos de retardo:

- *lectura/escritura de registro: 1 ut*
- *lectura o escritura del banco de registros: 5 ut*
- *retardo de la ALU: 55 ut*
- *decodificador: 4 ut*
- *buffer triestado: despreciable*
- *multiplexores: 2 ut*
- *acceso a Memoria de Control: 30 ut*
- *acceso a Memoria Principal: 100 ut*
- *secuenciador de microprograma: 15 ut*

b) Detalle a nivel RT (transferencia entre registros) la microsubrutina del ciclo de fetch.

c) Realice (a nivel RT) el microprograma de la fase de ejecución de la instrucción de una palabra:

`ADD #desp[.R5++], #desp[.R3++]`

SOLUCIÓN

a) La duración del ciclo de reloj debe dar tiempo a realizar la operación elemental más larga teniendo en cuenta que se actualice el estado, opere el secuenciador y lea en la memoria de control:

$T_{reloj} = t_{RC} + t_{dec} + t_R + t_{mux} + t_{ALU} + t_{SR} + t_{SR} + t_{SEC} + t_{RD} + t_{RD} + t_{MC} + t_{RC} =$
$= 1 + 4 + 1 + 2 + 55 + 1 + 1 + 15 + 1 + 1 + 30 + 1 = 113\ ut$

La operación de memoria principal necesitaría de 2 ciclos de reloj.

b) El incremento del PC debe hacerse a través de algún registro temporal conectado a la entrada de la ALU, y se debe incrementar en 8 unidades, ya que es direccionable a nivel de byte y la palabra tiene 64 bits. Para facilitar la suma del dato inmediato 8, se utiliza una microsubrutina que genera un 8 y lo deja en el registro temporal T3. Antes de incrementar el PC, hay que llamar a esta microsubrutina.

```
f1:  PC → T1, AR; μcall gen8T3
f2:  M(AR) → DR, T1+T3 → PC; si Wait ir a f2
f3:  DR→ IR, ir a CO
```

La microrutina gen8T3:

```
g1:  0+1 → T1;
g2:  T1x2+1 → T1;
g3:  T1x2+1 → T1;
g4:  T1x2+1 → T3; μret
```

c) Los dos operandos de la suma están en memoria, así como el destino, por lo que habrá que hacer tres accesos a memoria, con un microbucle de sincronización. Conviene acceder en último lugar al operando cuya dirección coincide con la dirección del destino, para que esté su dirección en AR antes de escribir en memoria el resultado. Como en el ciclo de fetch, al incrementar los registros R5 y R3 se llama antes a la microsubrutina que deja un 8 en el registro temporal T3. Hay que actualizar el SR en la microinstrucción que realiza la suma de los dos operandos de la instrucción. El microprograma de la fase de ejecución de la instrucción es el siguiente:

```
a1:   IR(desp) → T1
a2:   BR(R3) → T2
a3:   T2+T1 → AR; μcall gen8T3
a4:   M(AR) → DR, T2+T3 → BR(R3); si Wait ir a a4
a5:   DR → T4
a6:   IR(desp) → T1
a7:   BR(R5) → T2
a8:   T2+T1 → AR, T2+T3 → BR(R5)
a9:   M(AR) → DR; si Wait ir a a9
a10:  DR → T1
a11:  T4 → T2
a12:  T1+T2 → DR; actualizar SR
a13:  DR → M(AR); si Wait ir a a13
a14:  ir a fetch
```

2 ***(5 puntos)*** *Un computador cuenta con los siguientes formatos de representación:*

Formato 1. *Coma fija: 16 bits en complemento a uno con la coma entre los bits 3 y 4, por ejemplo 010001101110,1001.*

Formato 2. *Coma flotante: 16 bits, el bit superior es el signo, los siete siguientes el exponente, representado en exceso a 63 y los 8 siguientes la magnitud de la mantisa, que está representada en signo magnitud, con bit implícito y la coma a la derecha de este.*

a) Determine el rango y la resolución de ambos formatos.

b) Dada la representación de coma fija A = H'FDF1 determine su valor decimal y represente el número en el formato de coma flotante.

c) Dada la representación de coma flotante B = H'3F43 determine su valor decimal y represente el número en el formato de coma fija.

d) Realice la operación A-B en el formato 1, dejando el resultado en dicho formato, y determine su valor decimal.

e) Realice paso a paso la operación A+B en el formato 2, dejando el resultado en el formato de partida, y determine su valor decimal. En este caso de utilice para la operación dos bits de guarda y un bit retenedor, así como redondeo al más próximo.

SOLUCIÓN

a) Rango y resolución de los formatos.

Formato 1.

$$\begin{cases} \texttt{100000000000,0000} = \texttt{-011111111111,1111} & \rightarrow & -(2^{11} - 2^{-4}) \\ \texttt{011111111111,1111} & \rightarrow & 2^{11} - 2^{-4} \end{cases}$$

$Rango = [-(2^{11} - 2^{-4}), 2^{11} - 2^{-4}]$

$Resolucion = 2^{-4}$

Formato 2.

Exponente: $[-63, 64]$. Como hay que reservar el exponente -63 para la representación del cero, el rango del exponente queda: $[-62, 64]$.

$$\text{Mantisa: } \pm \begin{cases} \texttt{1,00000000} & \rightarrow & 1 \\ \texttt{1,11111111} & \rightarrow & 2 - 2^{-8} \end{cases}$$

$Rango = \pm [1 \cdot 2^{-62}, (2 - 2^{-8}) \cdot 2^{64}] \cup 0$

$Resolucion = 2^{-8} \cdot 2^{E}$

b) Valor decimal y representación en coma flotante.

$A = \texttt{H'FDF1} = \texttt{111111011111,0001} = \texttt{-000000100000,1110} = \texttt{-32,875}$

$A = \texttt{-1,00000111} \cdot 2^5 = \texttt{1 1000100 00000111} = \texttt{H'C407}$

c) Valor decimal y representación en coma fija.

$B = \texttt{H'3F43} = \texttt{0 0111111 01000011} = \texttt{+1,01000011} \cdot 2^0 = \texttt{1,26172}$

$B = \texttt{000000000001,0100} = \texttt{H'0014}$

d) $A - B$ en coma fija.

A	111111011111,0001
$+B$	111111111110,1011
$A + B$	1 111111011101,1100
EAC	000000000000,0001
$A + B$	111111011101,1101

$A + B = \texttt{H'FDDD} = \texttt{-000000100010,0010} = \texttt{-34,125}$

e) $A + B$ en coma flotante.

Los números a sumar son:

$A = \texttt{-100000111} \cdot 2^5 \qquad B = \texttt{+1,01000011} \cdot 2^0$

La diferencia de exponentes: $E_A - E_B = 5 - 0 = 5$ nos dice que hay que desplazar la mantisa de B cinco lugares a la derecha. Si partimos de:

$B = \texttt{+1,01000011} \cdot 2^0$

Al desplazar cinco lugares (teniendo en cuenta el bit retenedor) queda:

$B = \texttt{+0,00001010 00 1} \cdot 2^5$

Se realiza la suma de mantisas (resta):

M_A	$-\ 1,00000111\ 00\ 0$	
$+M_B$	$+\ 0,00001010\ 00\ 1$	
$A+B$	$-\ 0,11111100\ 11\ 1 \cdot 2^5$	
$A+B$	$-\ 1,11111001\ 11\ 0 \cdot 2^4$	*Normalizacion*
	$+\ \ \ \ \ \ \ \ \ \ \ \ \ \ 1$	*Redondeo*
$A+B$	$-\ 1,11111010\ 01\ 0 \cdot 2^4$	

$A+B$ = `-1,11111010` $\cdot 2^4$ = `1 10000011 11111010` = `H'C3FA`

$A+B$ = `-11111,1010` = `-31,625`

NOTAS: 5 de junio de 2013 DURACIÓN: 1 hora y 45 minutos

REVISIÓN: 6 de junio de 2013 PUNTUACIÓN: Especificada en cada problema

1 *Indique, justificando su respuesta, si las siguientes afirmaciones son verdaderas o falsas:*

a) La Entrada/Salida programada tiene la ventaja de que es la mejor cuando la CPU tiene que comunicarse con muchos periféricos, porque ella tiene el control total de la operación.

Falso. La E/S programada exige mayor utilización de la CPU, por lo que no es adecuada en el caso de tener muchos periféricos.

a) Los parámetros característicos de la memoria son: número de ciclos por instrucción, capacidad, y tamaño de los registros

Falso. El número de ciclos por instrucción y tamaño de los registros no son parámetros de la memoria. Sí lo son, además de la capacidad, el tiempo de acceso, el ancho de palabra.

a) El modelo Von Neumann se basa en una única memoria para almacenar tanto datos como instrucciones, por lo que en principio, se podría erróneamente intentar ejecutar un dato.

Verdadero. Al almacenar tanto datos como instrucciones, en el caso de que la memoria no se organice correctamente puede erróneamente intentarse un dato como si fuera una instrucción. Hay que establecer mecanismos adecuados para que eso no ocurra.

a) La Unidad de Control no necesita como entrada el registro PC, contador de programa.

Verdadero. La Unidad de Control necesita, entre otras entradas, la instrucción a ejecutar, que está en el registro de instrucción.

2 *Sea un computador de dos direcciones con modelo de ejecución registro-registro con palabra de 32 bits y direccionamiento a nivel de byte que dispone únicamente de direccionamiento inmediato, directo a registro e indirecto a registro. Realice los programas correspondientes a las instrucciones que se muestran a continuación para el computador indicado. Si es necesario utilice los registros RT1, RT2 y RT3 como registros auxiliares.*

```
XOR .R1,[.R4],#6[.R7]

     MOVE .R7,.RT1
     ADD .RT1,#6
     LD .R1,[.RT1]
     LD .RT2,[.R4]
     XOR .R1, .RT2
```

```
CALL [#3[.R7--]]

     MOVE .R7,.RT1
     ADD .RT1,#3
     LD .RT1,[.RT1]
     SUB .R7,#4
     CALL [.RT1]
```

```
ADD /2000, /1000 , [#16[++.R7]]

     LD .RT1,#2000
     LD .RT2,#1000
     LD .RT2,[.RT2]
     ADD .R7,#4
     MOVE .R7,.RT3
     ADD .RT3,#16
     LD .RT3,[.RT3]
     LD .RT3,[.RT3]
     ADD .RT2,.RT3
     ST .RT2,[.RT1]
```

3 *(5 puntos)*

Un programa en ensamblador del 88110 trabaja con una lista de n elementos enteros de una palabra. La lista está almacenada en posiciones consecutivas de memoria y no contiene elementos repetidos. El programa utiliza dos subrutinas para comprobar si la lista está ***ordenada en forma creciente*** *y, si no lo está, ordenarla.*

Debe programar las dos subrutinas que se describen a continuación. Supondrá que están definidas todas las macros que se han explicado en la parte teórica de la asignatura y que están reservados para tareas específicas los registros r1, r30(SP) y r31(FP). La pila crece hacia direcciones decrecientes de memoria y el puntero de pila apunta a la última posición ocupada de la cima de la pila.

a) Subrutina `ORDEN(lista, elementos)` *que compara cada elemento de la lista con el siguiente, empezando por el primero, para determinar si están ordenados de forma creciente. El paso de parámetros se realiza en la pila por dirección, siendo* `lista` *la dirección del primer elemento y* `elementos` *la dirección del número n de elementos. En el registro r29 se devuelve 0 si la lista está vacía y 1 cuando la lista está completamente ordenada. Cuando encuentra dos elementos no ordenados, llama a la subrutina* `INTERCAMBIO`*. Al retornar a la subrutina* `ORDEN` *se vuelve a empezar la comparación desde el primer elemento de la lista.*

b) Subrutina `INTERCAMBIO(menor)` *que recibe en la pila el parámetro* `menor`*, que corresponde a la dirección de un elemento de la lista y realiza el intercambio entre éste y el elemento de la posición de memoria inmediatamente anterior.*

SOLUCIÓN

a) Para programar la subrutina ORDEN se usan las macros PUSH y POP y se crea un marco de pila. Puesto que en el retorno se comienza de nuevo la comparación desde el primer elemento, se salvaguardan en la pila los registros r22 y r4 que corresponden a la carga inicial de la dirección de la lista y del número, n, de elementos de la lista. Los registros r3 y r21 se utilizan como contador de elementos y puntero a la lista, respectivamente. Cuando la lista está ordenada se devuelve un 1 en el registro r29, se destruye el marco de pila y se retorna al programa llamante.

```
ORDEN:          PUSH (r1)               ; salvaguardar dirección de retorno
                PUSH (r31)              ; salvaguardar FP antiguo
                or r31, r30, r0         ; crear marco de pila
                ld r20, r31, 12         ; r20 dirección del número de elementos
                ld r4, r20, r0          ; r4 número de elementos
                ld r22, r31, 8          ; r22 dirección de la lista
                cmp r5, r4, r0          ; ver si lista vacía
                bb1 eq, r5, vacia
empezar:        or r3, r4, r0           ; r3 contador de elementos
                or r21, r22, r0         ; r21 puntero a la lista
comparar:       subu r3, r3, 1          ; decrementar contador de elementos
                cmp r5, r3, r0          ; ver si no hay más elementos
                bb1 eq, r5, ordenada
                ld r6, r21, r0          ; cargar un elemento
                ld r7, r21, 4           ; y el siguiente
                ; si no están ordenados en forma creciente, este es el menor
                addu r21, r21, 4        ; incrementar puntero a lista
                cmp r5, r6, r7          ; ver si elementos ordenados
                bb1 lo, r5, comparar    ; si están ordenados, seguir comparando
                PUSH (r22)              ; salvaguarda de registros en la pila
                PUSH (r4)
                PUSH (r21)              ; pasar parámetro (dirección del elemento menor)
                bsr INTERCAMBIO         ; salto a subrutina
                addu r30, r30, 4        ; saltar parámetro
                POP (r4)                ; recuperar registros de la pila
                POP (r22)
                br empezar              ; empezar a comparar desde el primer elemento
vacia:          or r29, r0, r0          ; devolver que es vacía
                br volver
ordenada:       or r29, r0, 1           ; devolver que ordenada
```

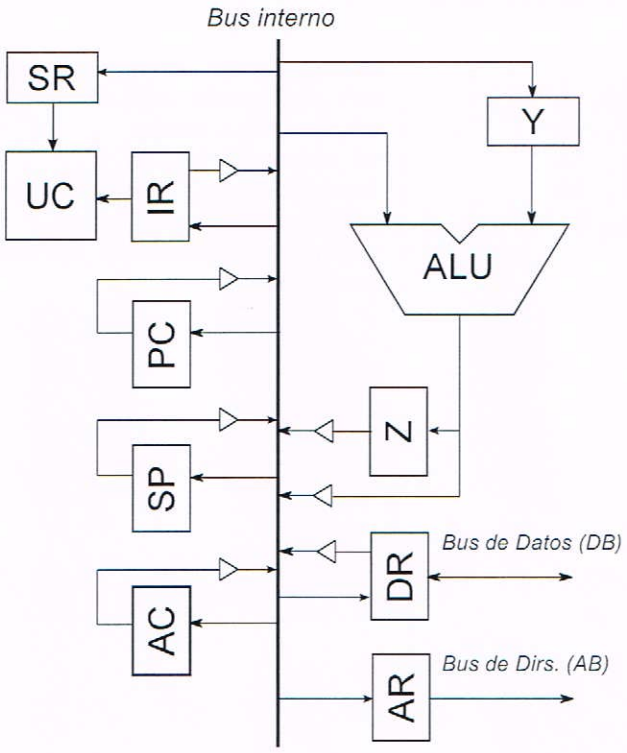
```
volver:     or r30, r31, r0      ; destruir marco de pila
            POP (r31)            ; recuperar FP
            POP (r1)             ; recuperar dirección de retorno
            jmp (r1)             ; retornar
```

b) En la programación de la subrutina INTERCAMBIO, no es necesario crear un marco de pila. Realiza el intercambio del contenido de dos posiciones de memoria.

```
INTERCAMBIO:  PUSH(r1)           ; salvaguardar dirección de retorno
              ld r21, r30, 4     ; r21 dirección del elemento menor
              ld r8, r21, r0     ; r8 elemento menor
              ld r9, r21, -4     ; r9 elemento mayor
              st r8, r21, -4     ; intercambiar, primero el menor
              st r9, r21, 0      ; y luego el mayor
              POP (r1)           ; recuperar dirección de retorno
              jmp (r1)           ; retornar
```

1 ***(5 puntos)*** *Sea la CPU cuyo esquema simplificado aparece en la figura. La ALU, todos los registros, rutas de datos y de direcciones son de 32 bits.*

PC: Reg. contador de programa	*SP: Reg. puntero de pila*
AR: Reg. de direcciones	*DR: Reg. de datos*
IR: Reg. de instrucciones	*SR: Reg. de estado*
Y, Z: registros transparentes	*AC: es el único registro de* propósito general



a) Suponiendo que:

- *las instrucciones ocupan todas una sola palabra (incluso las que usan un direccionamiento directo a memoria, /Dir, e inmediato, #valor).*
- *todas la microoperaciones duran un ciclo de reloj, incluidas las de acceso a memoria.*
- *el tiempo de ciclo se ha establecido es 100 ns.*
- *la ALU realiza las operaciones habituales (suma, resta, incremento, decremento, etc.)*
- *la pila crece hacia direcciones decrecientes y SP apunta a la primera dirección libre.*

a.1) Realice la descomposición en una secuencia de microoperaciones, indicando claramente las acciones que se realizan en cada ciclo de reloj, para:

1) el **fetch** *(común a todas las instrucciones, ya que todas ocupan una palabra.)*

2) las siguientes seis instrucciones (señale con "fetch" la secuencia anterior, supuesta al principio de cada instrucción). En cada caso, indique claramente el ciclo (de haberlo) en que se deba actualizar el registro de estado.

```
1) LD .AC, /Dir

2) ST .AC, /Dir

3) ADD .AC, #valor
```

4) BR /Dir

5) CALL /Dir

6) RET

a.2) *Según el apartado anterior, indique en cada caso el número total de ciclos –incluido el fetch– que tardaría en ejecutarse cada instrucción y su equivalente en tiempo. Calcule asimismo el número medio de ciclos y tiempo medio, supuesta una distribución uniforme de las seis instrucciones.*

b) *En una segunda fase se ha decidido que las microoperaciones que impliquen acceso a memoria ocupen dos ciclos de reloj y el tiempo de ciclo se baje a 60 ns. Para esta nueva situación, se pide:*

b.1) *Realice las modificaciones que estime necesarias en las descomposición en microoperaciones del apartado a).*

b.2) *Vuelva a calcular el número de ciclos y el tiempo de ejecución de cada una de las instrucciones y sus valores medios.*

b.3) *Comente los resultados obtenidos ahora en comparación con los que se obtuvieron anteriormente.*

SOLUCIÓN

a)

a.1) Descomposición en secuencias de microoperaciones:

1) **fetch:**

```
i:     AR ← PC
       Y ← PC
i+1:   DR ← M(AR)
       PC ← Y+1
i+2:   IR ← DR
```

2) instrucciones:

```
1) LD .AC, /Dir
          fetch
    i+3:  AR ← IR.Dir
    i+4:  DR ← M(AR)
    i+5:  AC ← DR
2) ST .AC, /Dir
          fetch
    i+3:  AR ← IR.Dir
    i+4:  DR ← AC
    i+5:  M(AR) ← DR
3) ADD .AC, #valor
          fetch
    i+3:  Y ← IR.valor
    i+4:  Z ← Y + AC; Actualizar SR
    i+5:  AC ← Z
4) BR /Dir
           fetch
    i+3:  PC ← IR.Dir
5) CALL /Dir
          fetch
    i+3:  DR ← PC
    i+4:  AR ← SP
          Y ← SP
    i+5:  M(AR) ← DR
          SP ← Y - 1
    i+6:  PC ← IR.Dir
6) RET
```

```
        fetch
    i+3:  Y ← SP
    i+4:  SP ← Y + 1
          AR ← Y + 1
    i+5:  DR ← M(AR)
    i+6:  PC ← DR
```

a.2) En cada caso el número total de ciclos que tardaría en ejecutarse cada instrucción y su equivalente en tiempo es:

```
fetch: 3 ciclos, 300 ns
LD: 6 ciclos, 600 ns
ST: 6 ciclos, 600 ns
ADD: 6 ciclos, 600 ns
BR: 4 ciclos, 400 ns
CALL: 7 ciclos, 700 ns
RET: 7 ciclos, 700 ns
```

El número medio de ciclos sería (6 * 3 + 4 + 7 * 2) ciclos / 6 inst = 6 ciclos / inst, equivalente a 600 ns / inst.

b)

b.1) La modificación introducida afectará directamente a las microoperaciones que realizan acceso a memoria y pudiera ser que implicasen una nueva optimización de la secuencia de microoperaciones de la instrucción correspondiente. Se observa que se realiza accesos a memoria en el fetch y en las instrucciones 1, 2, 5 y 6. Analizaremos cada uno de los casos.

1) **fetch**:

```
i:          AR ← PC
            Y ← PC
i+1, i+2:   DR ← M(AR)
 i+1:       PC ← Y+1
i+3:        IR ← DR
```

2) instrucciones:

```
1) LD .AC, /Dir
                fetch
    i+4:        AR ← IR.Dir
    i+5, i+6:   DR ← M(AR)
    i+7:        AC ← DR
2) ST .AC, /Dir
                fetch
    i+4:        AR ← IR.Dir
    i+5:         DR ← AC
    i+6, i+7:   M(AR) ← DR
5) CALL /Dir
                fetch
    i+4:        DR ← PC
    i+5:        AR ← SP
                Y ← SP
    i+6, i+7:   M(AR) ← DR
     i+6:       SP ← Y - 1
    i+7:        PC ← IR.Dir
6) RET
                 fetch
    i+4:        Y ← SP
    i+5:        SP ← Y + 1
                AR ← Y + 1
    i+6, i+7:   DR ← M(AR)
    i+8:        PC ← DR
```

Se observa que la modificación introduce un ciclo adicional en todas las instrucciones debido al fetch. En el caso de las instrucciones 1 y 2, añade otros ciclo más, y la dependencia de microoperaciones no permite ningún tipo de optimización adicional. También se añade una ciclo extra en las instrucciones 5 y 6, sólo en el caso de

la primera, la última microoperación es compatible con el segundo ciclo del acceso a memoria, por lo que se puede simultanear con éste.

b.2) De acuerdo con el apartado anterior y el nuevo tiempo de ciclo –60 ns–, se obtiene el siguiente número de ciclos y tiempo de ejecución de cada una de las instrucciones, así como sus valores medios.

```
fetch: 4 ciclos, 240 ns
LD: 8 ciclos, 480 ns
ST: 8 ciclos, 480 ns
ADD: 7 ciclos, 420 ns
BR: 5 ciclos, 300 ns
CALL: 8 ciclos, 480 ns
RET: 9 ciclos, 540 ns
```

El número medio de ciclos sería (8 * 3 + 5 + 9 + 7) ciclos / 6 inst = 7,5 ciclos / inst, equivalentes a 450 ns / inst.

b.3) Se comprueba que la descomposición de las microoperaciones más lentas en dos ciclos de reloj ha permitido reducir el tiempo de ciclo en un 40 %, de 100 a 60 ns, y aunque ha aumentado el número medio de ciclos por instrucción, 7,5 ciclos vs. los 6 originales, el tiempo medio de ejecución de las instrucciones se ha reducido en un 25 %, de los 600 anterior a los 450 ns que se obtienen ahora.

2 ***(5 puntos)*** *Un computador representa números de coma flotante con un formato de 16 bits que sigue las convenciones del estándar IEEE754 en todo excepto en los tamaños, es decir, usa el mismo tipo de representaciones especiales, representación del exponente, bit implícito, situación de la coma, representaciones normalizadas y no normalizadas, así como bits de guarda y redondeo. En el formato, el bit superior corresponde al signo, los siete siguientes al exponente y los ocho últimos a la magnitud de la mantisa.*

a) Determine el rango de representación y la resolución del formato.

b) Determine el valor decimal de los siguientes números que están representados en el formato:

$$A = H'4405 \quad B = H'BE04 \quad C = H'802C \quad D = H'7F00$$

c) Represente en el formato los números decimales:

$$E = 9 \cdot 2^{-68} \quad F = +205,2$$

d) Realice paso a paso la suma $A + B$ dejando el resultado en el formato de almacenamiento en memoria.

e) Determine el error absoluto cometido en la operación anterior.

SOLUCIÓN

1. **Rango y resolución del formato.**

   Números normalizados.

   Exponente: El estándar IEEE754 representa los exponentes en exceso a $2^{n-1} - 1$. En este caso sería exceso a 63. Como se reserva el exponente mínimo (−63) para la representación del cero y los números no normalizados, y el exponente máximo (+64) para la representación del infinito y las indeterminaciones, el rango de exponente para la representación de números queda: $[-62, 63]$.

   La mantisas normalizadas en este formato se representan en signo-magnitud, con bit implícito y la coma situada a la derecha del bit implícito:

   $$\text{Mantisa: } \pm \begin{cases} 1,00000000 & \rightarrow \quad 1 \\ 1,11111111 & \rightarrow \quad 2 - 2^{-8} \end{cases}$$

   El rango para números normalizados es: $\pm \left[1 \cdot 2^{-62}, (2 - 2^{-8}) \cdot 2^{63}\right]$

   Números no normalizados

Exponente: Siguiendo el estándar IEEE754, aunque los números no normalizados se representan con el exponente todo a ceros (valor -63), todos ellos tienen como exponente el más pequeño de los normalizados, en este caso -62. De este modo hay continuidad en la representación.

Mantisas: $\pm \begin{cases} 0,00000000 \rightarrow 0 \\ 0,00000000 \rightarrow 2^{-8} \\ 0,11111111 \rightarrow 1-2^{-8} \end{cases}$

El rango para números no normalizados es: $\pm\,[2^{-8}\cdot 2^{-62}, (1-2^{-8})\cdot 2^{-62}] \cup 0$

Así pues, el rango total es:

$\pm\,[1\cdot 2^{-62}, (2-2^{-8})\cdot 2^{63}] \cup \pm\,[2^{-8}\cdot 2^{-62}, (1-2^{-8})\cdot 2^{-62}] \cup 0$

La resolución depende del exponente y es: $2^{-8}\cdot 2^{E}$

2. **Valor decimal de los números.**

   $A = \texttt{H'4405} = \texttt{0 1000100 00000101} = \texttt{+1,00000101}\cdot 2^{5} = \texttt{+100000,101} = \texttt{+32,625}$

   $B = \texttt{H'BE04} = \texttt{1 0111110 00000100} = \texttt{-1,00000100}\cdot 2^{-1} = \texttt{-0,10000010} = \texttt{-0,5078125}$

   $C = \texttt{H'802C} = \texttt{1 0000000 00101100}$

   Como se puede ver, el exponente del número C es cero, con lo cual representa un número no normalizado, al que se le asigna el exponente mínimo (-62).

   $C = \texttt{-0,00101100}\cdot 2^{-62} = \texttt{-1011}\cdot 2^{-68} = \texttt{-3,73}\cdot 10^{-20}$

   $D = \texttt{H'7F00} = \texttt{0 1111111 00000000} = +\infty$

3. **Paso al formato.**

   $E = 9\cdot 2^{-68} = 1001\cdot 2^{-68}$

   No se puede representar como un numero normalizado porque el exponente queda fuera de rangoo. Lo representamos como un numero no normalizado. Buscamos el exponente -6:

   $E = \texttt{0,00100100}\cdot 2^{-62} = \texttt{0 0000000 00100100} = \texttt{H'0024}$

   $F = 205,2 = \texttt{11001101,0011} = \texttt{1,10011010}\cdot 2^{7} = \texttt{0 1000110 10011010} = \texttt{H'469A}$

4. **Suma $A + B$.**

   Este formato utiliza dos bits de guarda y un bit retenedor para la suma. Los números a sumar son:

   $A = \texttt{+1,00000101}\cdot 2^{5}$ *y* $B = \texttt{-1,00000100}\cdot 2^{-1}$

   Se restan los exponentes: $E_A - E_B = 5 - (-1) = 6$. Hay que desplazar la mantisa de B seis lugares a la derecha. Así, teniendo en cuenta los dos bits de guarda y el bit retenedor queda:

   $B = \texttt{-0,00000100 00 1}\cdot 2^{5}$

M_A	1,00000101 00 0	
M_B	− 0,00000100 00 1	
	$\texttt{1,00000000 11 1}\cdot 2^{5}$	*Normalizado*
Redondeo	0,00000000 10 0	
	$\texttt{1,00000001 01 1}\cdot 2^{5}$	

   Representación:

   $A + B = \texttt{+1,00000001}\cdot 2^{5} = \texttt{0 1000100 00000001} = \texttt{H'4401}$

5. **Error absoluto.**

   $A + B = 32,625 - 0,5078125 = 32,1171875$

   $A + B = \texttt{+1,00000001}\cdot 2^{5} = 100000,001 = 32,125$

   $Error = \|32,1171875 - 32,125\| = 0,0078$

1 *(1 punto) Sea una cinta magnética con las siguientes características:*

- *Tiempo de arranque y de parada: 3 ms.*
- *Claros IRG de 0,1 cm de longitud.*
- *Densidad de grabación lineal: 80.000 bits/mm.*
- *Bloques de 20.000 bytes.*

Suponiendo que en el tiempo t=0 la cinta se encuentra en movimiento a la velocidad lineal de escritura y que en escribir 2 bloques consecutivos y detener la cinta se emplean 4,25 ms,

a) calcule cuál es la velocidad de transferencia de la unidad de cinta.

SOLUCIÓN

$$\frac{(20.000+(80.000\times1)/8+20.000)bytes}{V_{transf}} + 3\times10^{-3}s = 4,25\times10^{-3}s$$

$$V_{transf} = \frac{50.000bytes}{1,25\times10^{-3}s} = 40\times10^{6}\ bytes/s$$

2 *Sea un teléfono móvil cuya pantalla gráfica tiene una resolución de 480 × 640 píxeles con una gama de 65.536 colores y frecuencia de refresco de 50 Hz cuya placa gráfica está estropeada y debe reemplazar. Teniendo en cuenta la situación de crisis económica en que nos encontramos,*

a) (1 punto) justifique cuál de los siguientes modelos debe comprar para reparar el teléfono:

1.- Memoria gráfica de 1 Mbyte, palabras de 32 bits y tiempo de acceso de 40 ns. Precio: 150 euros.

2.- Memoria gráfica de 2 Mbytes, palabras de 16 bits y tiempo de acceso de 20 ns. Precio: 175 euros.

3.- Memoria gráfica de 2 Mbytes, palabras de 16 bits y tiempo de acceso de 100 ns. Precio: 125 euros.

4.- Memoria gráfica de 1 Mbyte, palabras de 32 bits y tiempo de acceso de 100 ns. Precio: 125 euros.

En el teléfono móvil se ejecuta una aplicación que captura el contenido de la pantalla y lo envía por su conector miniusb usando una transmisión en serie asíncrona de 500.000 bits por segundo, con 8 bits de dato.

b) (1 punto) Si la transmisión de la pantalla emplea 12,288 segundos, determine cuál es el formato de las tramas de datos empleado en la transmisión en serie.

SOLUCIÓN

a) Memoria y ancho de banda requeridos por la pantalla:

$(480\times640)\times\frac{log_2 65.536}{8}\ bytes = 614.400\ bytes$

$614.400\ bytes\times50\ Hz = 30.720.000\ bytes/s$

Memoria y ancho de banda proporcionados por las placas gráficas:

1.- 1 Mbyte, $\frac{(32/4)bytes}{40\times10^{-9}s} = 100.000.000\ bytes/s$

2.- 2 Mbyte, $\frac{(16/4)bytes}{20\times10^{-9}s} = 100.000.000\ bytes/s$

3.- 2 Mbyte, $\frac{(16/4)bytes}{100\times10^{-9}s} = 20.000.000\ bytes/s$

4.- 1 Mbyte, $\frac{(32/4)bytes}{100\times10^{-9}s} = 40.000.000\ bytes/s$

Las placas gráficas nº 1, 2 y 4 satisfacen los requisitos de la placa gráfica. Teniendo en cuenta su precio, la placa que se debe instalar es la nº 4.

b) Usando una velocidad de 500.000 bits/s, en 12,288 segundos se transmiten $500.000\times12,288 = 6.144.000$ bits.

Por cada dato se transmiten $6.144.000/614.400 = 10$ bits. Dado que se transmiten 8 bits de dato, sólo restan 2 bits para los bits de start, stop y paridad. Por tanto, el formato de transmisión es de 8 bits de dato, sin paridad y con un bit de stop.

50

3 *Sea una unidad de disco duro de brazo móvil con las siguientes características:*

- *$102{,}40512 \cdot 10^9$ bytes de capacidad neta.*
- *20.001 cilindros, 10 superficies y 500 sectores por pista.*
- *Velocidad de rotación: 12.000 rpm.*
- *Velocidad de transferencia 128 Mbytes/s ($128 \cdot 10^6$ bytes/s).*
- *El tiempo que emplea en mover la cabeza de un cilindro a otro consecutivo es 0,01 ms.*
- *El tiempo de estabilización de las cabezas es 1 ms.*

*a) (**1 punto**) Calcule la capacidad bruta de la unidad de disco.*

*b) (**6 puntos**) En el instante t=0 s la cabeza se encuentra situada al comienzo del sector absoluto 4.483.044. En ese instante se ordena la escritura de un fichero que ocupa 514.048 bytes. El comienzo del fichero se almacena en los sectores 3.982.944 y 4.984.645; el resto, en sectores consecutivos a partir de este último.*

b.1) Calcule el instante en el que finaliza la escritura de dicho fichero.

b.2) Calcule en qué instante finalizaría la escritura suponiendo que el resto del fichero se almacenara en sectores distribuidos aleatoreamente por el disco.

SOLUCIÓN

a) Capacidad bruta:

Capacidad Bruta de cada pista = Vtransf/rps = $\frac{128 \cdot 10^6 \ bytes/s}{12.000 rpm/60 \ s/m}$ = 640.000 bytes.

Capacidad Bruta = 640.000 × 20.001 × 10 = 128.006.400.000 bytes

b) Coordenadas de los sectores:

4.483.044/(10 × 500) = 896

$4.483.044 mod(10 \times 500) = 3.044$

3.044/500 = 6

$3.044 mod(500) = 44$

4.483.044: (896, 6, 44).

3.982.944: (796, 5, 444).

4.984.645: (996, 9, 145).

Tiempo de giro: 1/(12.000/60) rps = 0,005 s = 5 ms. Tiempo de avance de un sector: 5/500 = 0,01 ms.

Lectura del sector 3.982.944:

Tbúsqueda = (896 - 796)0,01 + 1 = 2 ms.
Avanza 2/0,01 = 200 sectores, situándose delante del sector 44 + 200 = 244.
TLatencia = (444 - 244)0,01 = 2 ms.
Lectura del primer sector: 2 ms + 2 ms + 0,01 ms = 4,01 ms (queda situado delante del sector 445).

Lectura del sector 4.984.645:

Tbusqueda = (996 - 796)0,01 + 1 = 3 ms.
Avanza 3/0,01 = 300 sectores, situándose delante del sector (445 + 300)mod(500) = 245.
TLatencia = (500 - 245 + 145)0,01 = 4 ms.
Lectura del segundo sector: 3 ms + 4 ms + 0,01 ms = 7,01 ms (queda situado delante del sector 146).

Número de sectores del fichero:

Tamaño de 1 sector $\frac{102{,}40512 \cdot 10^9 \ bytes}{20.001 \times 10 \times 500} = 1.024 \ bytes$

N° de sectores = $\frac{514.048 \ bytes}{1.024 \ bytes} = 502$

b.1) A partir del segundo sector, los 500 restantes tienen un tiempo de acceso nulo, salvo que se produzca un cambio de cilindro. Los primeros 500 - 146 = 354 pertenecen a la última superficie del cilindro 996, empleándose $354 \times 0,01$ ms = 3,54 ms en su lectura. Los siguientes y últimos 146 pertenecen a la primera superficie del cilindro 997. El cambio de pista provoca un tiempo de acceso al primero de estos sectores de 5 ms: 1,01 ms de tiempo de búsqueda y 3,99 ms de tiempo de latencia, girando el disco una vuelta completa. Por tanto, el tiempo empleado en la lectura de los últimos sectores es de 5 ms + $146 \times 0,01$ ms = 6,46 ms.

Tiempo total: $4,01 + 7,01 + 3,54 + 6,46 = 21,02\ ms$

b.2) Tiempo medio de acceso:

$T_{macc} = \frac{(20.001-1)0,01}{2} + 1 + \frac{5}{2} = 103,5\ ms$

Tiempo total: $4,01 + 7,01 + 500 \times (103,5 + 0,01) = 51.766,02\ ms$

1 (2 puntos) *Indique, justificando su respuesta, si las siguientes afirmaciones son verdaderas o falsas:*

a) La Entrada/Salida por DMA necesita que la CPU sincronice la operación de E/S.

b) La Unidad de Control no necesita como entrada el registro de estado (RE).

c) El registro de direcciones de memoria es un registro transparente al usuario, ya que este no tiene por qué utilizarse en las instrucciones máquina.

d) La velocidad de la memoria se indica por el número de MIPS.

SOLUCIÓN

a) Falso. La entrada salida mediante acceso directo a memoria (DMA) es la que proporciona mejor rendimiento ya que es el periférico el que realiza tanto la sincronización como la transferencia de la información. La CPU debe iniciar la operación, y al final de ésta atender una interrupción para conocer que ha finalizado la operación.

b) Falso. Para la unidad de control el RE es un registro fundamental conocer, puesto que debe ejecutar instrucciones de salto condicionales y debe conocer la condición. Además, en el RE se almacenan otras condiciones por las que debe estar pendiente la unidad de control (errores).

c) Verdadero El registro de direcciones de memoria es un registro utilizado por el procesador para estabilizar la dirección que se envia al bus de direcciones externo, y que el usuario no puede utilizar explícitamente en las instrucciones.

d) Falso. MIPS es un indicador de velocidad de la CPU (millones de instrucciones ejecutadas por segundo). La velocidad de la memoria se indica por su velocidad y/o por su tiempo de acceso y tamaño de palabra.

2 (8 puntos) *La tabla* `ALUMNOS` *contiene la información de los alumnos que cursan una asignatura. Su primer campo es un entero de 4 bytes que contiene el identificador del alumno. Tras ese campo, cada entrada tiene* `n` *campos de enteros con las notas de evaluaciones parciales. Una nota con valor* `0xffffffff` *indica No Presentado a esa prueba. Tras los campos de notas, hay un último campo que contiene un entero con la nota media ponderada del alumno y que inicialmente contiene el valor* `0xffffffff`*. Cada entrada de la tabla tiene, por lo tanto,* `n+2` *campos de enteros de 4 bytes. El fin de la tabla está indicado con el valor* `0xffffffff` *en el campo de identificador del alumno.*

La tabla `PESOS` *contiene los* `n` *pesos a aplicar a las notas parciales. Estos pesos son enteros de 0 a 10.*

Se desea actualizar el último campo de la tabla ALUMNOS con las notas medias ponderadas según la tabla `PESOS`*. Para ello debe realizar:*

a) la función `nota_media=MEDIA_POND (n, Notas, Pesos)` *que calcula la nota ponderada de los* `n` *elementos del vector* `Notas`*, considerando los* `n` *pesos del vector* `Pesos`*.* `n` *se pasa por valor, y* `Notas` *y* `Pesos` *por dirección. El resultado se pasa por valor en el registro r29. Deberá tener en cuenta que no puntúen las pruebas no presentadas.* $r29 = \sum_i nota_i * peso_i$ *siendo* $nota_i \geq 0$

b) la subrutina `CALCULAR_SEMESTRE (n, ALUMNOS, Tabla_Pesos)` *que recibe el número* `n` *de notas parciales de la asignatura (valor) y dos tablas pasadas por dirección. La subrutina calcula el último campo de todas las entradas de la tabla* `ALUMNOS` *con la nota media ponderada (según* `Tabla_Pesos`*) de las* `n` *calificaciones parciales. Para ello llamará a la subrutina del apartado anterior* `MEDIA_POND`*. Se recomienda guardar en una variable local el puntero en curso a entrada de la tabla* `ALUMNOS`

Se supondrá que están definidas todas las macros que se han explicado en la parte teórica de la asignatura, que la subrutina llamante deja disponibles todos los registros excepto r1, r30 (SP) y r31 (FP); que la pila crece hacia direcciones de memoria decrecientes y el puntero de pila apunta a la última posición ocupada (de la misma forma que se ha utilizado en el proyecto) y que todos los parámetros de las subrutinas anteriores se pasan en la pila.

A modo de ejemplo, a continuación se muestra una situación inicial de las tablas `ALUMNOS` *y* `PESOS` *y el resultado en la tabla* `ALUMNOS` *para* `n` $= 3$ *notas parciales.*

`ALUMNOS`

id_alumno	*nota1*	*nota2*	*nota3*	*media*
1	*10*	*10*	*10*	*-1*
2	*10*	*-1*	*-1*	*-1*
3	*0*	*10*	*5*	*-1*
-1	*??*	*??*	*??*	*??*

`PESOS`

4	*4*	*2*

La tabla resultado es:

ALUMNOS

id_alumno	*nota1*	*nota2*	*nota3*	*media*
1	*10*	*10*	*10*	*100*
2	*10*	*-1*	*-1*	*40*
3	*0*	*10*	*5*	*50*
-1	*??*	*??*	*??*	*??*

SOLUCIÓN

a) La función media_pond debe recorrer dos vectores de n elementos, por lo que se usan sendos punteros (r20 y r21), que se inicializan al valor de los parámetros. El recorrido consistirá en un bucle que se ejecuta un número determinado de veces (r5) y se pasa como parámetro. El resultado de la función se debe dejar en el registro r29, que se inicializa con el valor 0, y servirá para ir acumulando el sumatorio de notas por pesos. Si una nota indica no presentado, no se acumula al sumatorio y se avanza a la siguiente.

```
media_pond: PUSH(r1)
        ld r5,r30,4         ; n
        ld r20,r30,8        ; dir tabla de notas
        ld r21,r30,12       ; dir tabla de pesos
        xor r29, r29, r29
buclem: cmp r2,r5,r0
        bb1 eq, r2, fin
        ld r10,r20,r0       ; en r10 guarda nota
        ld r11,r21,r0       ; en r11 guarda peso
        cmp r2, r10, -1     ; si no presentado no sumar
        bb1 eq, r2, avanza
        mulu r10, r10, r11
        addu r29, r29, r10
avanza: addu r20,r20,4
        addu r21,r20,4
        subu r5, r5, 1
        br buclem
fin:    POP(r1)
        jmp(r1)
```

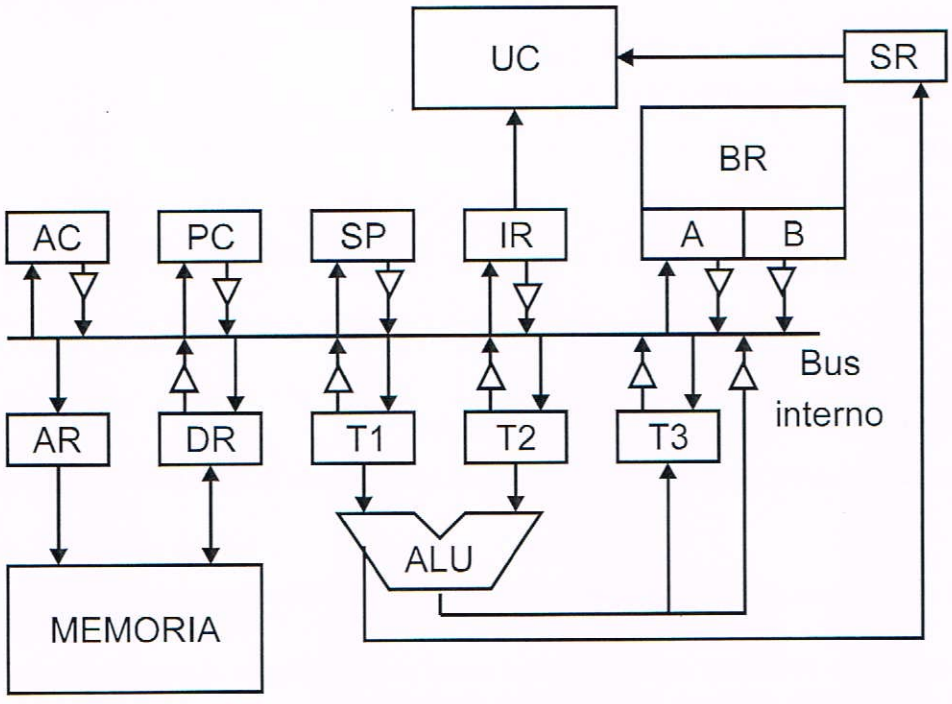
b) La solución de la rutina calcular_semestre crea el marco de pila para contener una variable local (puntero en curso a la entrada de la tabla alumnos). El puntero a la entrada en curso de la tabla de alumnos ocupa el desplazamiento -4 al puntero de marco r31 y se inicializa con la dirección de comienzo de la tabla de alumnos.

Se recorre la tabla de alumnos hasta fin de lista. Para cada entrada se llama a la subrutina media_pond del apartado anterior para que calcule la nota. La nota calculada que la subrutina devuelve en r29 se almacena en la tabla de alumnos en el último campo.

```
calcular_semestre: PUSH(r1)
        PUSH(r31)
        or r31,r30,r30
        subu r30,r30,4      ; espacio para la variable local
        ld r22,r31,12       ; puntero a tabla alumnos
        ld r21,r31,16       ; puntero a tabla pesos
        st r22,r31,-4       ; var. local puntero en curso a alumnos
buclec: ld r3,r22,r0        ; id_alumno de tabla alumnos
        cmp r5,r3,-1
        bb1 eq, r5, fin_buc
        addu r20, r22, 4    ; r20 comienzo a notas del alumno
        PUSH(r21)
        PUSH(r20)
        ld r6, r31, 8       ; parámetro n
```

```
        PUSH(r6)
        bsr media_pond
        addu r30, r30, 12   ; sacar parámetros de pila
        ld r22, r31, -4     ; leer variable local
        ld r6, r31, 8       ; guardar nota en r22+(n+1)*4
        addu r6, r6, 1
        mulu r6, r6, 4
        st r29,r22, r6
        addu r6, r6, 4      ; Se avanza al siguiente alumno
        addu r22,r22,r6     ; r22+(n+2)*4
        st r22, r31, r-4     ; guarda variable local
        br buclec
fin_buc: or r30,r31,r31
        POP(r31)
        POP(r1)
        jmp(r1)
```

1 ***(3,5 puntos)*** *En la figura se muestra el esquema de un computador de 32 bits con Unidad de control microprogramada y direccionamiento a nivel de palabra, cuya memoria es síncrona. Los incrementos o decrementos de los registros, se realizan a través de la ALU.*



a) Calcule el mínimo tiempo del ciclo de reloj del computador, considerando los siguientes valores de tiempo para cada uno de los componentes indicados:

- *lectura/escritura de registro: 1 ut*
- *lectura o escritura del banco de registros: 5 ut*
- *retardo de la ALU: 35 ut*
- *decodificador: 4 ut*
- *buffer triestado: despreciable*
- *multiplexores: 2 ut*
- *acceso a Memoria de Control: 20 ut*
- *acceso a Memoria Principal: 100 ut*
- *secuenciador de microprograma: 8 ut*

b) Realice a nivel RT el microprograma de la instrucción de dos palabras, perteneciente al juego de instrucciones de este computador: `SUB #desp[.R2],/dir`. *La segunda palabra de la instrucción contiene exclusivamente la dirección del direccionamiento absoluto directo a memoria.*

c) Calcule cuánto tarda en ejecutarse dicha instrucción

SOLUCIÓN

a) El tiempo de ciclo se calcula:

$$Tck = T_r + T_{ALU} + T_{tri} + T_{RE} + T_{Sec} + T_{uPC} + T_{MControl} + T_{RC} = 1 + 35 + 1 + 8 + 1 + 20 + 1 = 67\ ut$$

Puesto que el tiempo de acceso a la memoria principal es de 100 ut, el acceso a la misma se realizará en dos ciclos.

b) El microprograma se incluye a continuación comenzando por el ciclo de fetch:

```
f1:  PC → T1, AR
f2:  M(AR) → DR; T1+1 → PC
f3:  M(AR) → DR
f4:  DR→ IR; ir a CO
```

Los dos operandos de la resta están en memoria, así como el destino, por lo que habrá que hacer cuatro accesos a memoria: uno para conseguir la segunda palabra de la instrucción, otro para conseguir el segundo

operando de la resta, otro para conseguir el primer operando y el último para escribir el resultado. Conviene acceder en último lugar al operando cuya dirección coincide con la dirección del destino, para que esté su dirección en AR antes de escribir en memoria el resultado. Hay que actualizar el SR en la microinstrucción que realiza la resta de los dos operandos de la instrucción. El microprograma de la fase de ejecución de la instrucción es el siguiente:

```
a1:   PC → T1, AR
a2:   M(AR) → DR; T1+1 → PC
a3:   M(AR) → DR
a4:   DR → AR
a5:   M(AR) → DR; R2 → T1
a6:   M(AR) → DR; IR(desp) → T2
a7:   T1 + T2 → AR
a8:   DR → T2; M(AR) → DR
a9:   M(AR) → DR
a10:  DR → T1
a11:  T1 - T2 → DR; actualizar SR
a12:  DR → M(AR)
a13:  DR → M(AR); PC → T1, AR; ir a f2(fetch+1)
```

c) La instrucción emplea 4 ciclos para el fetch y 13 para la ejecución de la instrucción, es decir, 17 ciclos en total.

2 ***(5 puntos)*** *La ALU de un computador trabaja con los dos formatos de representación que se indican a continuación:*

Formato 1. Coma fija*: 16 bits en signo-magnitud con el primer bit para el signo y los siguientes 15 bits para la magnitud (6 bits de parte entera y 9 bits de parte fraccionaria).*

Formato 2. Coma flotante*: 16 bits, con el bit superior para el signo, los seis siguientes para el exponente (en exceso a 32) y los 9 siguientes para la magnitud de la mantisa, que está representada en signo-magnitud, con bit implícito y la coma a la izquierda de éste.*

1. *Determine el rango y la resolución de ambos formatos.*
2. *Dados los números decimales A = 60,5 y B = -7,8125 represéntelos en ambos formatos.*
3. *Realice paso a paso y justificadamente la operación A-B en ambos formatos, expresando finalmente el resultado en el formato de partida. En el caso de coma flotante utilice dos bits de guarda, un bit retenedor y redondeo al más próximo.*
4. *Especifique el error producido al realizar cada una de las operaciones anteriores.*

SOLUCIÓN

a) Rango y resolución de ambos formatos.

Formato 1

$$\pm \begin{cases} 000000,000000001 & \rightarrow & 2^{-9} \\ 111111,111111111 & \rightarrow & 2^{6} - 2^{-9} \end{cases}$$

$Rango = \pm\left[(2^6 - 2^{-9}), 2^{-9}\right] \cup 0$

$Resolucion = 2^{-9}$

Formato 2

Exponente: $[-32, 31]$. Se reserva exponente -32 para la representación del cero.

$$\text{Mantisa: } \pm \begin{cases} ,1000000000 & \rightarrow & 2^{-1} \\ ,1111111111 & \rightarrow & 1 - 2^{-10} \end{cases}$$

$Rango = \pm\left[(1 - 2^{-10}) \cdot 2^{31}, 2^{-1} \cdot 2^{-31}\right] \cup 0$

$Resolucion = 2^{-10} \cdot 2^E$

b) Representación de los números A y B en los formatos 1 y 2.

$A = 60,5 =$ 111100,1

$B = -7,8125 =$ -111,1101

$A(formato1) =$ 0 111100 100000000

$B(formato1) =$ 1 000111 110100000

$A(formato2) =$,1111001 $\cdot 2^6 =$ 0 100110 111001000

$B(formato2) =$ -,1111101 $\cdot 2^3 =$ 1 100011 111101000

c) Operación $A - B$ en los formatos 1 y 2.

Formato 1

Los números a restar son: $A =$ 0 111100 100000000 y $B =$ 1 000111 110100000

La operación A-B se convierte en una suma, cambiando el signo del sustraendo: A+(-B). Puesto que los signos de los sumandos son iguales, el signo del resultado es el signo común y la operación a realizar es la suma de las magnitudes:

$$\begin{array}{r} 111100,100000000 \\ +\ 000111,110100000 \\ \hline 1000100,010100000 \end{array}$$

Se observa un acarreo final que indica un error por desbordamiento.

Formato 2

Los números a restar son: $A =$,1111001000 $\cdot 2^6$ y $B =$ -,1111101000 $\cdot 2^3$

Como los exponentes son distintos, el exponente del resultado es: E = EA = 6.

Se restan los exponentes: $E_A - E_B = 3$.

Hay que desplazar la mantisa de B tres lugares a la derecha, utilizando dos bits de guarda y un bit retenedor:

$B =$ -,0001111101 00 0 $\cdot 2^6$

Puesto que los signos de A y de -B son iguales, el signo del resultado es el signo común: S = 0.

La operación a realizar es MA + MB.

$$\begin{array}{rll} & ,1111001000 & 00\ 0 \\ +\ & ,0001111101 & 00\ 0 \\ \hline & 1,0001000101 & 00\ 0 \\ & ,1000100010 & 10\ 0 \\ +\ & & 1 \\ \hline & ,1000100011 & \end{array}$$

No normalizada.
Normalización.
Redondeo
Normalizada.

Al haber desplazado la mantisa una posición a la derecha, por postnormalización, aumenta el exponente, luego: $A - B =$,1000100011 $\cdot 2^7 = 1000100,011 = 68,375$.

En el formato 2: $A - B =$ 0 100111 000100011

d) Error producido en las operaciones.

Formato 1

El resultado de la operación es erróneo, ya que se produce desbordamiento por excederse la capacidad de representación de este formato. Se activará el correspondiente bit de desbordamiento del registro de estado.

Formato 2

El error absoluto producido en la operación es:

Error= 68,375-68,3125 = 0,0625

ESTRUCTURA DE COMPUTADORES (GradoII)
CONVOCATORIA EXTRAORDINARIA DE JULIO (1 de Julio de 2014)

PROBLEMAS

1 **(3 puntos)** *Programe las siguientes subrutinas:*

a) **(10 %)** *La función indicada a continuación proporciona la dirección del elemento* `(i,j)` *de la* `matriz` *cuadrada de orden* `n`*:* `dir_elemento = dir_elem(i,j,matriz,n)`

Todos los parámetros de pasan en la pila, `i`*,* `j` *y* `n` *son parámetros que se pasan por valor, teniendo en cuenta que los elementos de la matriz están en el rango [0,n-1].* `matriz` *indica la dirección de comienzo de la matriz. Esta subrutina devuelve en el registro r29 la dirección de memoria indicada en los párrafos anteriores.*

b) **(90 %)** *La función indicada a continuación intercambia la diagonal principal con la diagonal secundaria de una* `matriz` *cuadrada de orden* `n`*.* `intercambia_diagonales(matriz,n)`*. Al igual que en la subrutina anterior todos los parámetros se pasan en la pila y esta subrutina puede hacer uso de la subrutina anterior.*

A modo de ejemplo se muestra la ejecución de una llamada a `intercambia_diagonales` a la que se pasa como parámetro la matriz que aparece en la figura (de orden 4) y el resultado de la ejecución se indica a la derecha de este párrafo.

$$Matriz = \begin{pmatrix} \mathbf{1} & 2 & 3 & \mathbf{4} \\ 5 & \mathbf{6} & \mathbf{7} & 8 \\ 9 & \mathbf{10} & \mathbf{11} & 12 \\ \mathbf{13} & 14 & 15 & \mathbf{16} \end{pmatrix}$$

$$Resultado = \begin{pmatrix} \mathbf{4} & 2 & 3 & \mathbf{1} \\ 5 & \mathbf{7} & \mathbf{6} & 8 \\ 9 & \mathbf{11} & \mathbf{10} & 12 \\ \mathbf{16} & 14 & 15 & \mathbf{13} \end{pmatrix}$$

Para la realización de este ejercicio se llevará a cabo el tratamiento del Marco de Pila descrito en clase y se supondrá que están definidas todas las macros que se han explicado en la parte teórica de la asignatura; que las subrutinas llamantes dejan disponibles todos los registros excepto r1, r30 (SP) y r31 (FP); que la pila crece hacia direcciones de memoria decrecientes y el puntero de pila apunta a la última posición ocupada (de la misma forma que se ha utilizado en el proyecto).

SOLUCIÓN

a) Para calcular la dirección de un elemento de una matriz cuadrada aplicaremos la fórmula:

$$dir\_elem(i,j) = dir\_matriz + (i \cdot n + j) \cdot 4$$

El código de la función se muestra a continuación:

```
dir_elem:
        ld r3,r30,r0            ; i
        ld r4,r30,12            ; n
        mulu r3,r3,r4           ; i * n
        ld r4,r30,4             ; j
        addu r3,r3,r4           ; i * n + j
        mulu r3,r3,4            ; (i * n + j) * 4
        ld r29,r30,8            ; matriz
        addu r29,r3,r29         ; matriz + (i * n + j) * 4
        jmp(r1)
```

b) Para realizar el intercambio de la matriz se realizará un bucle u se buscará en cada iteración el elemento (k,k) de la matriz y se intercambiará con el (k,n-k-1). La subrutina crea el marco de pila y utilizará dos variables locales:

- `k`: Es un entero que contiene el índice del elemento de la diagonal principal que se está evaluando. Ocupa la dirección `r31-4`.
- `tmp`: Es un entero que contiene la dirección del elemento de la diagonal principal que se está evaluando. Es necesario puesto que se va a realizar una llamada a `dir_elem` y la dirección se puede perder si está almacenada en un registro. Ocupa la dirección `r31-8`.

El código de la función se muestra a continuación:

```
intercambia_diag:
        PUSH(r1)
        PUSH(r31)
        or r31,r30,r30
        subu r30,r30,8
        or  r2,r0,r0            ; Contador elementos diagonal (k)
bucle:  ld r3,r31,12            ; n
        cmp r7,r2,r3            ; Si hemos acabado con la diagonal se va a fin
        bb1 eq,r7,fin
        st r2,r31,-4            ; k
        PUSH(r3)
        ld r20,r31,8            ; matriz
        PUSH(r20)
        PUSH(r2)                ; Se accede al elemento (k,k)
        PUSH(r2)
        bsr dir_elem
        addu r30,r30,16
        st r29,r31,-8           ; La dir del elemento (k,k) se almacena en una var. local
        ld r2,r31,-4
        ld r3,r31,12
        PUSH(r3)                ; n
        ld r20,r31,8            ; matriz
        PUSH(r20)
        subu r3,r3,r2
        subu r3,r3,1
        PUSH(r3)                ; Se accede al elemento (k,n-k-1)
        PUSH(r2)
        bsr dir_elem
        addu r30,r30,16
        ld r2,r29,r0
        ld r20,r31,-8
        ld r3,r20,r0
        st r3,r29,r0
        st r2,r20,r0
        ld r2,r31,-4
        addu r2,r2,1            ; k = k - 1
        br bucle
fin:    or r30,r31,r31
        POP(r31)
        POP(r1)
        jmp(r1)
```

2 *(2 puntos) Sea un procesador de 64 bits con unidad de control cableada, un solo bus interno de datos y direcciones y tres registros transparentes (TMP1, TMP2 y TMP3). La duración del ciclo de reloj es de 30 ut. La Memoria principal se direcciona a nivel de byte y su tiempo de acceso es de 100 ut. La pila crece en direcciones decrecientes y el puntero de pila apunta al primer hueco.*

a) Especifique a nivel RT (transferencia entre registros) las operaciones elementales que se realizan durante las fases de fetch y de ejecución de la siguiente instrucción de dos palabras:

```
PUSH [/dir]
```

b) Determine razonadamente el tiempo total que tardaría en ejecutarse la instrucción anterior.

SOLUCIÓN

La operación de memoria principal necesitaría de 4 ciclos de reloj.

a)

El incremento del PC debe ser de 8 unidades, por tener direccionamiento a nivel de byte y tener la palabra 64 bits. El desglose de o peraciones elementales para ejecutar la instrucción `CALL` puede ser el siguiente:

```
1:  PC → AR ; leer 2ª palabra
2:  M(AR) → DR ; PC+8 → PC
3:  DR → AR
4:  M(AR) → DR
5:  DR → AR
6:  M(AR) → DR
7:  SP → AR
8:  DR → M(AR); SP-8 → SP; ir a fetch guardar en pila, postdecrementar SP
```

El ciclo de fetch es:

```
f1:  PC → AR
f2:  M(AR) → DR ; PC+8 → PC
f3:  DR → RI; ir a C.O.
```

b)

$T_{instr} = t_{eje} + t_{fetch} = (8 + 3 \cdot 3) + (3 + 1 \cdot 3)\ c\ =\ 23\ c\ =\ 23\ c \cdot 30\ ut\ =\ 690\ ut$

3 **(3 puntos)** *Un computador cuenta con un formato de representación de coma flotante de 12 bits. El bit más significativo es el de signo, los cinco siguientes el exponente, representado en exceso a 16. El resto de los bits pertenecen a la mantisa, que está representada en signo magnitud, con bit implícito y la coma a la izquierda de éste. Su unidad aritmética, que solo cuenta con el operador de suma-resta,* **utiliza un bit de guarda y un bit retenedor**, *así como redondeo al más próximo.*

a) Determine el rango y la resolución del formato.

b) Dado el número decimal $A = +22,625$ represéntelo en el formato.

c) La cadena hexadecimal $H'BD8$ corresponde a un número representado en el formato. Determine su valor decimal.

d) Realice paso a paso la operación $A/2 + 3 \cdot B$, dejando el resultado en el formato.

e) Determine el error absoluto cometido en la operación anterior.

SOLUCIÓN

a) **Rango y resolución del formato.**

Exponente: $[-16, 15]$. Como hay que reservar el exponente -16 para la representación del cero, el rango del exponente queda: $[-15, 15]$.

56

Mantisa: $\pm \begin{cases} ,1000000 & \rightarrow & 2^{-1} \\ ,1111111 & \rightarrow & 1-2^{-7} \end{cases}$

$Rango = \pm\,[2^{-1}\cdot 2^{-15}, (1-2^{-7})\cdot 2^{15}] \cup 0$

$Resolucion = 2^{-7}\cdot 2^{E}$

b) Representación en el formato.

$A = \texttt{+22,625} = \texttt{+10110,1010} = \texttt{+,1011010}\cdot 2^5 = \texttt{0 10101 011010} = \texttt{H'55A}$

c) Valor decimal.

$B = \texttt{H'BD8} = \texttt{1 01111 011000} = \texttt{-,1011000}\cdot 2^{-1} = \texttt{-,01011000} = \texttt{-0,34375}$

d) $(A/2)+3\cdot B$ en coma flotante.

$A = \texttt{+,1011010}\cdot 2^5$ *y* $B = \texttt{-,1011000}\cdot 2^{-1}$

$A/2$ se obtiene decrementando en uno el exponente de A. Así: $A/2 = \texttt{+,1011010}\cdot 2^4$

$3\cdot B$ se obtiene sumando $2\cdot B + B$

$2\cdot B$ se obtiene incrementando en uno el exponente de B. Así hay que sumar:

$2\cdot B = \texttt{-,1011000}\cdot 2^0$ *y* $B = \texttt{-,1011000}\cdot 2^{-1}$

$E_{2B} - E_B = 1$ Luego hay que desplazar la mantisa de B un lugar a la derecha:

Se realiza la suma de mantisas:

M_{2B}	$-$,1011000 0 0	
$+M_B$	$-$,0101100 0 0	
$3B$	$-1,0000100\ 0\ 0\cdot 2^0$	
$3B$	$-$,1000010 0 0 $\cdot 2^1$	*Normalizacion*
	$-$ 1	*Redondeo*
$3B$	$-$,1000010 0 0 $\cdot 2^1$	

$3\cdot B = \texttt{-,1000010}\cdot 2^1$

$(A/2)+3\cdot B$. Diferecia de exponentes: $E_{A/2} - E_{3B} = 4-1 = 3$. Hay que desplazar la mantisa de 3B tres lugares a la derecha. Se realiza la suma de mantisas (resta):

$M_{A/2}$	+,1011010 0 0	
$+M_{3B}$	$-$,0001000 0 1	
$A/2+3B$	+,1010001 1 1 $\cdot 2^4$	*Normalizado*
	+ 1	*Redondeo*
	+,1010010 0 1 $\cdot 2^4$	

$(A/2)+3\cdot B = \texttt{+,1010010}\cdot 2^4 = \texttt{0 10100 010010} = \texttt{H'512}$

e) Error absoluto

$(A/2)+3\cdot B = \texttt{10,28125}$ $\quad (A/2)+3\cdot B = \texttt{1010,010} = \texttt{+10,25}$

$E_A = ||\texttt{10,28125 - 10,25}|| = \texttt{0,03125}$

4 **(2 puntos)** *Sea una unidad de disco duro con las siguientes características:*

- *12 superficies y 19.001 cilindros.*
- *Los sectores tienen un 80 % de información neta y un 20 % de información adicional para direccionamiento, CRCs, IRGs, etc.*
- *Velocidad de rotación: 12.000 rpm.*
- *Velocidad de transferencia: 64 MB/s ($64 \cdot 10^6$ bytes/s).*
- *Tiempo que emplea en mover la cabeza de una pista a otra consecutiva: 2 μs.*
- *Tiempo de estabilización de las cabezas: 1 ms.*

Dicha unidad de disco duro contiene un fichero de 307.200 bytes almacenado en sectores consecutivos a partir del sector 1.234.550 cuyas coordenadas geométricas son (411, 6, 50).

a) Calcule el tiempo medio de acceso.

b) Calcule la capacidad bruta.

c) En t=0 las cabezas de la unidad de disco se encuentran en el cilindro 911 al comienzo del sector 100. Calcule en qué instante terminarán las operaciones para la lectura del fichero.

Una vez finalizada la lectura del fichero, se envía éste por una línea serie a 38.400 bits/s en modo asíncrono con paridad par y un bit de stop.

d) Calcule la duración de la transmisión del fichero.

SOLUCIÓN

a)

$$Tiempo\ de\ rotación = \frac{60\ s/minuto}{12{,}000\ revoluciones/minuto} = 0{,}005\ s/revolución = 5\ ms/revolución$$

$$\bar{t}_{acc} = \bar{t}_{pos} + \bar{t}_{est} + \bar{t}_{lat} = \frac{19{,}001 - 1}{2} \times 0{,}002\ ms + 1\ ms + \frac{5\ ms}{2} = 19\ ms + 1\ ms + 2{,}5\ ms = 22{,}5\ ms.$$

b)

$$\begin{aligned} Capacidad\ bruta\ pista &= 64 \cdot 10^6\ bytes/s \times 5 \cdot 10^{-3}\ s/pista = 320{,}000\ bytes/pista \\ Capacidad\ bruta\ HD &= 12\ superficies \times 19{,}001\ pistas/superficies \times 320.000\ bytes/pista = \\ &= 72{,}963{,}840{,}000\ bytes \approx 73\ GB \end{aligned}$$

c)

$$\begin{aligned} \frac{1{,}234{,}550}{12\ pistas/cilindro \times X\ sectores/pista} &= 411 \\ X &= 1{,}234{,}55012 \times 411 = 250\ sectores/pista \end{aligned}$$

$$\begin{aligned} Capacidad\ bruta\ sector &= \frac{320{,}000\ bytes/pista}{250\ sectores/pista} = 1{,}280\ bytes/sector \\ Capacidad\ neta\ sector &= 1{,}280\ bytes/sector \times \frac{80}{100} = 1{,}024\ bytes/sector \\ Sectores\ fichero &= \frac{307{,}200\ bytes}{1{,}024\ bytes/sector} = 300\ sectores \end{aligned}$$

El fichero se extiende desde el sector (411, 6, 50) hasta el sector (411, 7, 99). Para alcanzar el primer sector habrá que avanzar 500 cilindros desde el cilindro 911 inicial.

$$\begin{aligned} t_{búsqueda} &= 500 \times 0,002\ ms + 1\ ms = 2\ ms \\ t_{sector} &= \frac{5\ ms/pista}{250\ sectores/pista} = 0,020\ ms/sector \\ Sectores\ recorridos\ durante\ el\ t_{búsqueda} &= \frac{2\ ms}{0,020\ ms/sector} = 100\ sectores \end{aligned}$$

Como la posición inicial era al comienzo del sector 100, tras haber girado 100 sectores las cabezas se encuentran al comienzo del sector 200. Para alcanzar el primer sector del fichero (411, 6, 50), hay que esperar un tiempo de latencia equivalente al giro de 100 sectores.

$$t_{latencia} = 100\ sectores \times 0,020\ ms/sector = 2\ ms$$

A continuación, se leerán los 200 sectores restantes de esa pista y 100 más de la siguiente.

$$Tiempo\ de\ lectura = 300\ sectores \times 0,020\ ms/sector = 6\ ms$$

La lectura acaba tras 2 ms + 2 ms + 6 ms = 10 ms.

d)

Para transmitir cada byte se utilizan 8 bits de datos, 1 bit de *start*, 1 bit de paridad y 1 bit de *stop*. Totalizando 11 bits por cada byte.

$$t_{transmisión} = \frac{307.200\ bytes \times 11\ bits/bytes}{38{,}400\ bits/s} = 88\ s$$